Daftar Isi

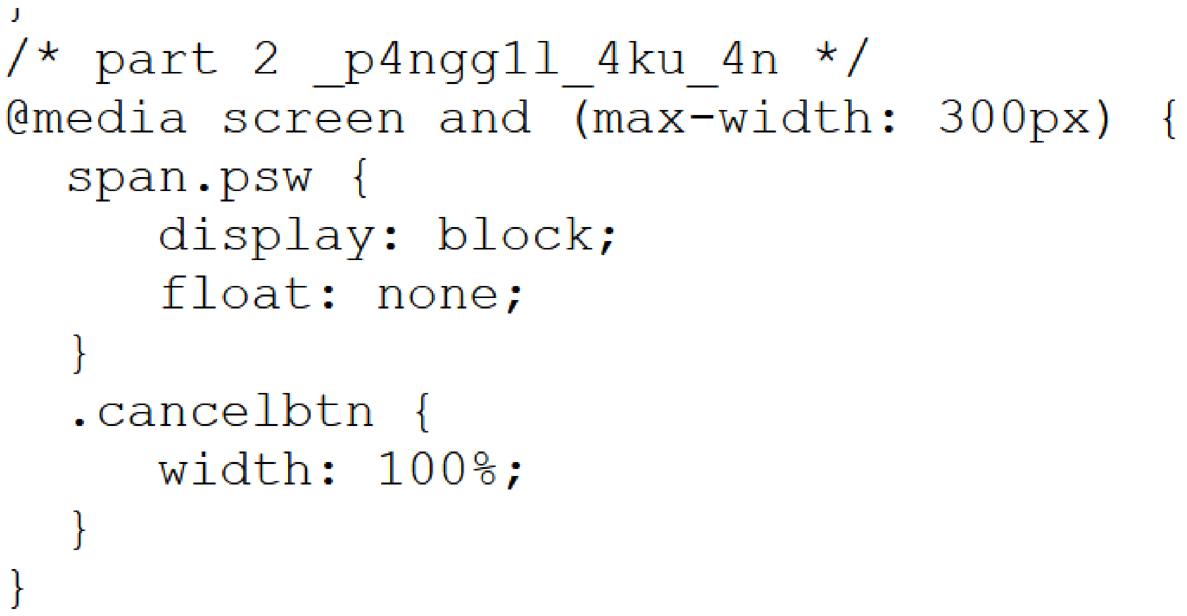
Web Exploitation

Ladu Singh

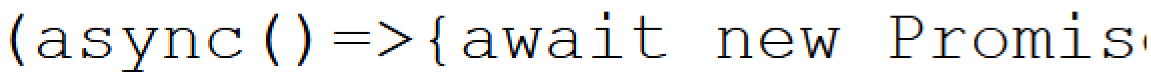
Langkah Penyelesaian:

Buka websitenya, check_source, liat di html, css, js

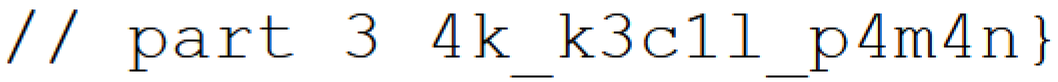
```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="/static/style.css">
<script src="/static/script.js"></script>
</head>
<body>
  <img width=500px src="/static/ladu-singh.jpg" alt="" class="center">
<!-- part 1 of the flag : CTFTED2021{j4ng4 -->
</body>
</html>
```



```
/* part 2 _p4ngg1l_4ku_4n */
@media screen and (max-width: 300px) {
  span.psw {
     display: block;
     float: none;
  }
  .cancelbtn {
     width: 100%;
  }
}
```



```
(async()=>{await new Promis
```



```
// part 3 4k_k3c1l_p4m4n}
```

Flag:

```
CTFTED2021{j4ng4_p4ngg1l_4ku_4n4k_k3c1l_p4m4n}
```

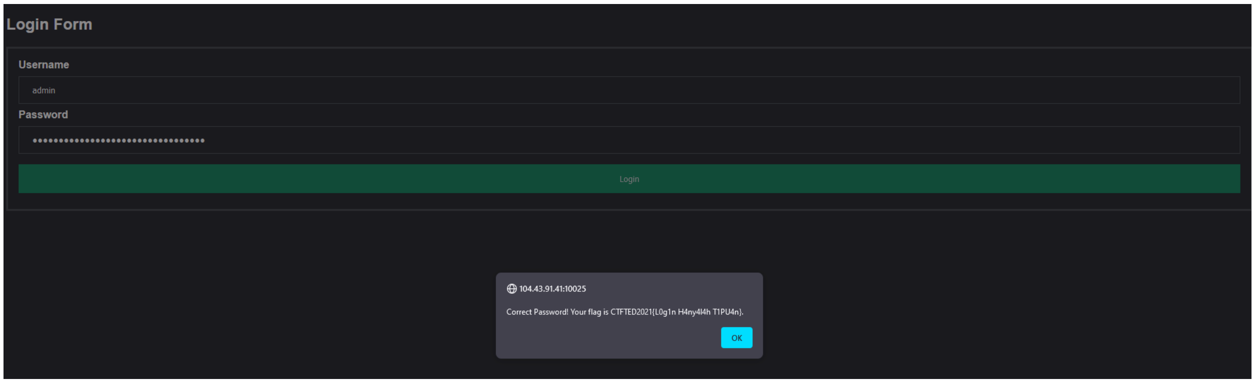
Login Bang

Langkah Penyelesaian:

Buka websitenya, check_source, buka js

```
return 'YWRtaW4' !== t.u ? alert('Incorrect Username') :
'Q1RGVEVEMjAyMXtMMGcxbiBING55NGw0aCBUMVBVNG59' !== t.p ?
alert('Incorrect Password') : void alert(`Correct Password!
Your flag is ${ atob(t.p) }.`)
```

Ada ini, tinggal base64 decode username sama passwordnya.
Passwordnya itu flagnya



Login Form

Username: admin

Password: •••••••••••••••••••••••••••••••••

104.43.91.41:10025

Correct Password! Your flag is CTFTED2021{L0g1n H4ny4l4h T1PU4n}.

Flag:

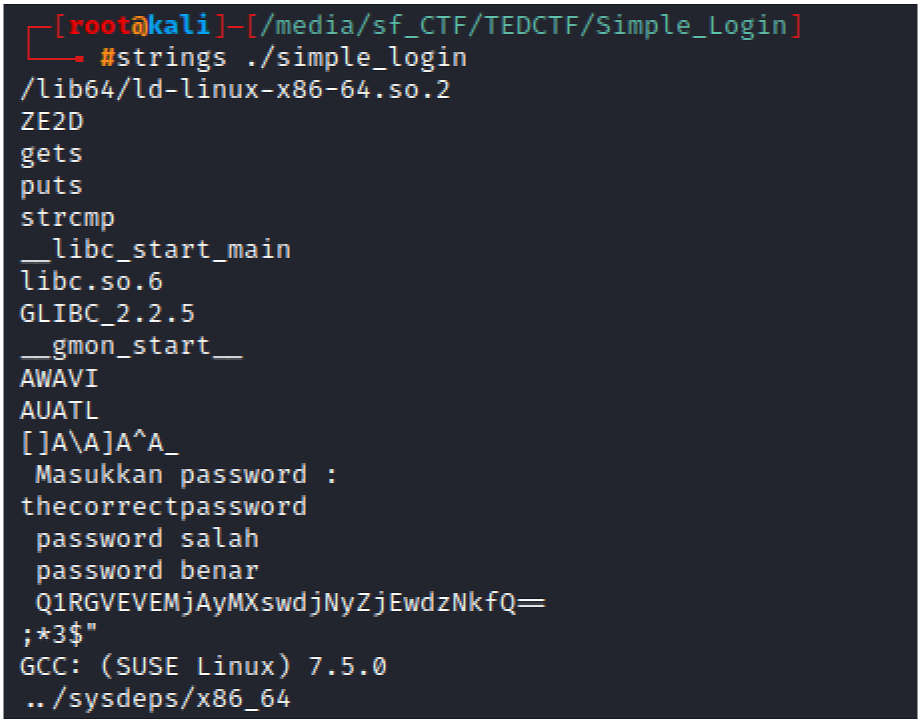
CTFTED2021{L0g1n H4ny4l4h T1PU4n}

Reverse Engineering

Simple Login

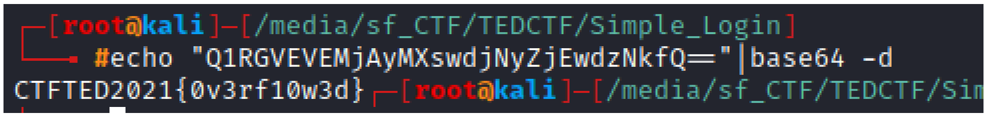
Langkah Penyelesaian:

Strings file ELFnya



```
┌─[root@kali]─[/media/sf_CTF/TEDCTF/Simple_Login]
└──╼ #strings ./simple_login
/lib64/ld-linux-x86-64.so.2
ZE2D
gets
puts
strcmp
__libc_start_main
libc.so.6
GLIBC_2.2.5
__gmon_start__
AWAVI
AUATL
[]A\A]A^A_
 Masukkan password :
thecorrectpassword
 password salah
 password benar
 Q1RGVEVEMjAyMXswdjNyZjEwdzNkfQ==
;*3$"
GCC: (SUSE Linux) 7.5.0
../sysdeps/x86_64
```

Saya melihat ada base64, langsung base64 decode



```
┌─[root@kali]─[/media/sf_CTF/TEDCTF/Simple_Login]
└──╼ #echo "Q1RGVEVEMjAyMXswdjNyZjEwdzNkfQ=="|base64 -d
CTFTED2021{0v3rf10w3d}┌─[root@kali]─[/media/sf_CTF/TEDCTF/Sim
```

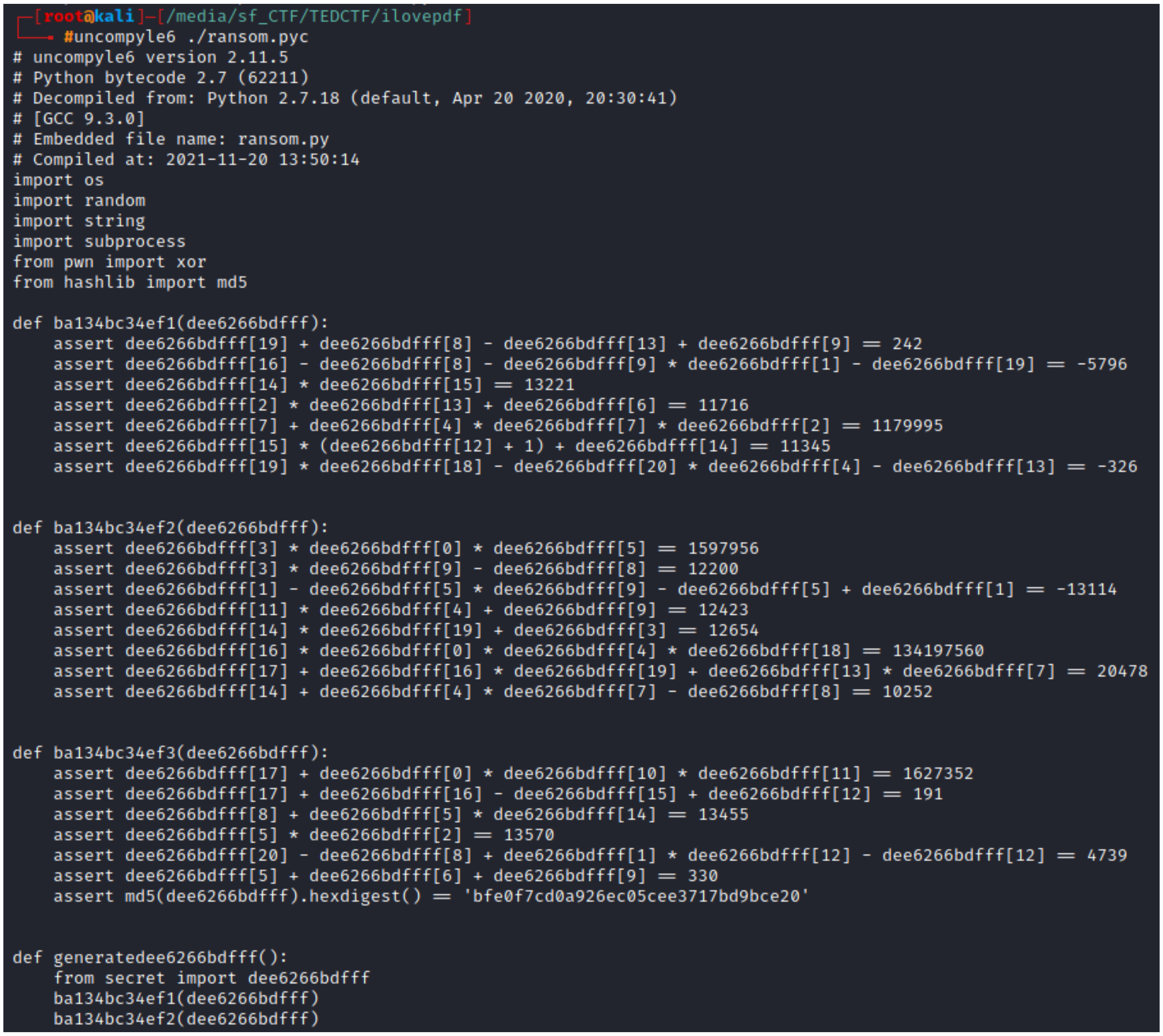
Flag:

CTFTED2021{0v3rf10w3d}

ilovepdf

Langkah Penyelesaian:

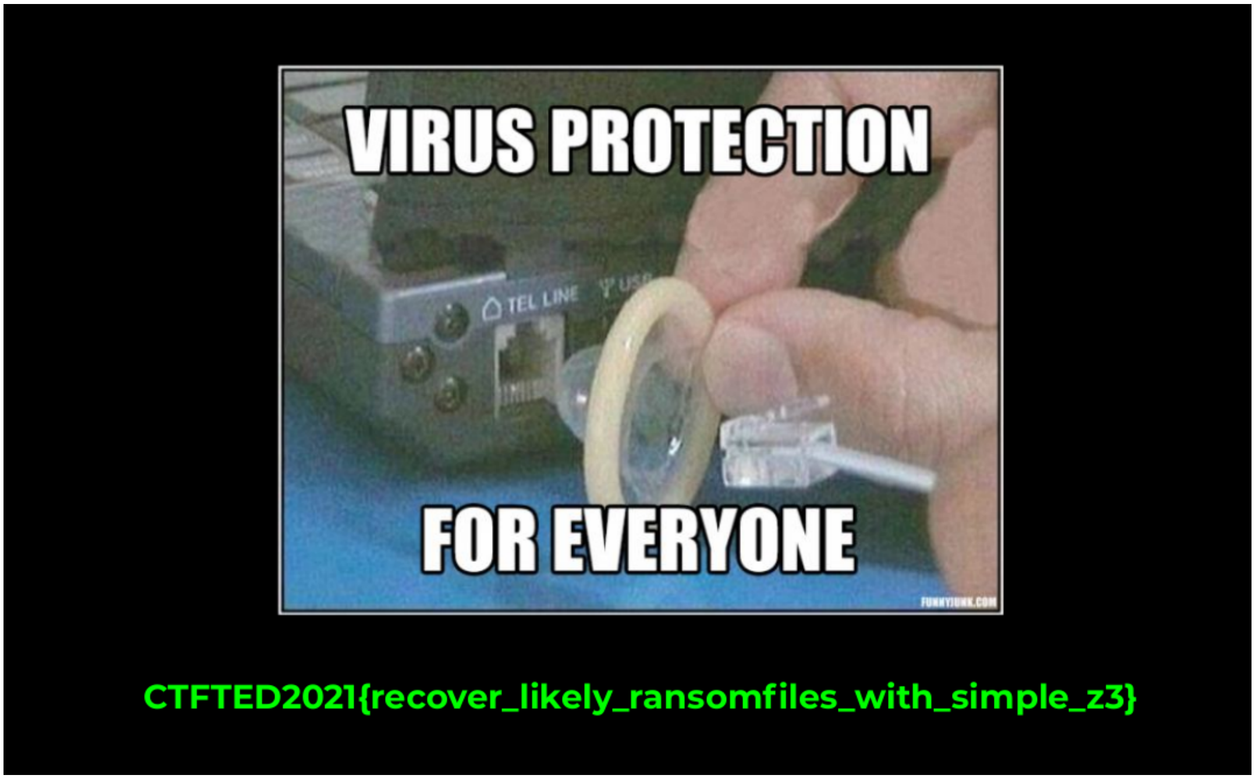
Saya melihat pyc, langsung pake uncompyle6



```
┌─[root@kali]─[/media/sf_CTF/TEDCTF/ilovepdf]
└──╼ #uncompyle6 ./ransom.pyc
# uncompyle6 version 2.11.5
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.18 (default, Apr 20 2020, 20:30:41)
# [GCC 9.3.0]
# Embedded file name: ransom.py
# Compiled at: 2021-11-20 13:50:14
import os
import random
import string
import subprocess
from pwn import xor
from hashlib import md5

def ba134bc34ef1(dee6266bdfff):
    assert dee6266bdfff[19] + dee6266bdfff[8] - dee6266bdfff[13] + dee6266bdfff[9] == 242
    assert dee6266bdfff[16] - dee6266bdfff[8] - dee6266bdfff[9] * dee6266bdfff[1] - dee6266bdfff[19] == -5796
    assert dee6266bdfff[14] * dee6266bdfff[15] == 13221
    assert dee6266bdfff[2] * dee6266bdfff[13] + dee6266bdfff[6] == 11716
    assert dee6266bdfff[7] + dee6266bdfff[4] * dee6266bdfff[7] * dee6266bdfff[2] == 1179995
    assert dee6266bdfff[15] * (dee6266bdfff[12] + 1) + dee6266bdfff[14] == 11345
    assert dee6266bdfff[19] * dee6266bdfff[18] - dee6266bdfff[20] * dee6266bdfff[4] - dee6266bdfff[13] == -326


def ba134bc34ef2(dee6266bdfff):
    assert dee6266bdfff[3] * dee6266bdfff[0] * dee6266bdfff[5] == 1597956
    assert dee6266bdfff[3] * dee6266bdfff[9] - dee6266bdfff[8] == 12200
    assert dee6266bdfff[1] - dee6266bdfff[5] * dee6266bdfff[9] - dee6266bdfff[5] + dee6266bdfff[1] == -13114
    assert dee6266bdfff[11] * dee6266bdfff[4] + dee6266bdfff[9] == 12423
    assert dee6266bdfff[14] * dee6266bdfff[19] + dee6266bdfff[3] == 12654
    assert dee6266bdfff[16] * dee6266bdfff[0] * dee6266bdfff[4] * dee6266bdfff[18] == 134197560
    assert dee6266bdfff[17] + dee6266bdfff[16] * dee6266bdfff[19] + dee6266bdfff[13] * dee6266bdfff[7] == 20478
    assert dee6266bdfff[14] + dee6266bdfff[4] * dee6266bdfff[7] - dee6266bdfff[8] == 10252


def ba134bc34ef3(dee6266bdfff):
    assert dee6266bdfff[17] + dee6266bdfff[0] * dee6266bdfff[10] * dee6266bdfff[11] == 1627352
    assert dee6266bdfff[17] + dee6266bdfff[16] - dee6266bdfff[15] + dee6266bdfff[12] == 191
    assert dee6266bdfff[8] + dee6266bdfff[5] * dee6266bdfff[14] == 13455
    assert dee6266bdfff[5] * dee6266bdfff[2] == 13570
    assert dee6266bdfff[20] - dee6266bdfff[8] + dee6266bdfff[1] * dee6266bdfff[12] - dee6266bdfff[12] == 4739
    assert dee6266bdfff[5] + dee6266bdfff[6] + dee6266bdfff[9] == 330
    assert md5(dee6266bdfff).hexdigest() == 'bfe0f7cd0a926ec05cee3717bd9bce20'


def generatedee6266bdfff():
    from secret import dee6266bdfff
    ba134bc34ef1(dee6266bdfff)
    ba134bc34ef2(dee6266bdfff)
```

Dari sini kita bersihkan variabelnya dan melihat ada 3 functions yang membantu kita untuk recover secretnya. Dari sana tinggal kita z3 untuk mendapatkan secretnya.

Setelah mendapatkan secretnya kita bisa mengembalikan secret2 dengan melakukan xor dengan signature pdf, dan rsecret dengan melakukan xor terhadap secret dan secret2.



CTFTED2021{recover_likely_ransomfiles_with_simple_z3}

Code:

ransom.py

```
# uncompyle6 version 2.11.5
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.18 (default, Apr 20 2020, 20:30:41)
# [GCC 9.3.0]
# Embedded file name: ransom.py
# Compiled at: 2021-11-20 13:50:14
import os
import random
import string
import subprocess
from pwn import xor
from hashlib import md5

def ba134bc34ef1(dee6266bdfff):
    assert dee6266bdfff[19] + dee6266bdfff[8] - dee6266bdfff[13] + dee6266bdfff[9] == 242
    assert dee6266bdfff[16] - dee6266bdfff[8] - dee6266bdfff[9] * dee6266bdfff[1] - dee6266bdfff[19] == -5796
    assert dee6266bdfff[14] * dee6266bdfff[15] == 13221
    assert dee6266bdfff[2] * dee6266bdfff[13] + dee6266bdfff[6] == 11716
    assert dee6266bdfff[7] + dee6266bdfff[4] * dee6266bdfff[7] * dee6266bdfff[2] == 1179995
    assert dee6266bdfff[15] * (dee6266bdfff[12] + 1) + dee6266bdfff[14] == 11345
    assert dee6266bdfff[19] * dee6266bdfff[18] - dee6266bdfff[20] * dee6266bdfff[4] - dee6266bdfff[13] == -326


def ba134bc34ef2(dee6266bdfff):
    assert dee6266bdfff[3] * dee6266bdfff[0] * dee6266bdfff[5] == 1597956
    assert dee6266bdfff[3] * dee6266bdfff[9] - dee6266bdfff[8] == 12200
    assert dee6266bdfff[1] - dee6266bdfff[5] * dee6266bdfff[9] - dee6266bdfff[5] + dee6266bdfff[1] == -13114
    assert dee6266bdfff[11] * dee6266bdfff[4] + dee6266bdfff[9] == 12423
    assert dee6266bdfff[14] * dee6266bdfff[19] + dee6266bdfff[3] == 12654
    assert dee6266bdfff[16] * dee6266bdfff[0] * dee6266bdfff[4] * dee6266bdfff[18] == 134197560
    assert dee6266bdfff[17] + dee6266bdfff[16] * dee6266bdfff[19] + dee6266bdfff[13] * dee6266bdfff[7] == 20478
    assert dee6266bdfff[14] + dee6266bdfff[4] * dee6266bdfff[7] - dee6266bdfff[8] == 10252


def ba134bc34ef3(dee6266bdfff):
    assert dee6266bdfff[17] + dee6266bdfff[0] * dee6266bdfff[10] * dee6266bdfff[11] == 1627352
    assert dee6266bdfff[17] + dee6266bdfff[16] - dee6266bdfff[15] + dee6266bdfff[12] == 191
    assert dee6266bdfff[8] + dee6266bdfff[5] * dee6266bdfff[14] == 13455
    assert dee6266bdfff[5] * dee6266bdfff[2] == 13570
    assert dee6266bdfff[20] - dee6266bdfff[8] + dee6266bdfff[1] *
dee6266bdfff[12] - dee6266bdfff[12] == 4739
    assert dee6266bdfff[5] + dee6266bdfff[6] + dee6266bdfff[9] ==
330
    # assert md5(dee6266bdfff).hexdigest() ==
'bfe0f7cd0a926ec05cee3717bd9bce20'


def generatedee6266bdfff():
    from secret import dee6266bdfff
    print dee6266bdfff
    print dee6266bdfff[16] - dee6266bdfff[8] - dee6266bdfff[9] *
dee6266bdfff[1] - dee6266bdfff[19]
    ba134bc34ef1(dee6266bdfff)
    ba134bc34ef2(dee6266bdfff)
    ba134bc34ef3(dee6266bdfff)
    dee6266bdfff_int = int.from_bytes(dee6266bdfff,
byteorder='big')
    for i in range(4):
        dee6266bdfff_int >>= dee6266bdfff[i * 4]
        dee6266bdfff_int <<= dee6266bdfff[i * 4]

    dee6266bdfff = dee6266bdfff_int.to_bytes(len(dee6266bdfff),
'big')
    return dee6266bdfff


def ransom(c651bca63aaas, dee6266bdfff):
    c651bca63aaa = open(c651bca63aaas, 'rb').read()
    rdee6266bdfff = os.urandom(len(dee6266bdfff))
    dee6266bdfff2 = xor(rdee6266bdfff, dee6266bdfff)
    a622337 = ''.join(random.choices(string.ascii_uppercase +
string.digits, k=5)).encode()
    w = open('ransom/broke_' + a622337 + '.pdf', 'wb+')
    w.write(os.urandom(1337))
    w.write(xor(c651bca63aaa[:5] * 5, dee6266bdfff2))
    w.write(xor(c651bca63aaa[5:], rdee6266bdfff))
    w.write(os.urandom(1337))
```

```
dee6266bdfff = generatedee6266bdfff()
print (dee6266bdfff)
baab3636 = subprocess.check_output('ls | grep .text',
shell=True).split('\n')[:-1]
for _ in baab3636:
    ransom(_, dee6266bdfff)
# okay decompiling ./ransom.pyc
```

recoverz3.py

```
from z3 import *
from hashlib import md5

secret = [Int(i) for i in range(21)]
s = Solver()
s.add(secret[19] + secret[8] - secret[13] + secret[9] == 242)
s.add(secret[16] - secret[8] - secret[9] * secret[1] - secret[19] == -5796)
s.add(secret[14] * secret[15] == 13221)
s.add(secret[2] * secret[13] + secret[6] == 11716)
s.add(secret[7] + secret[4] * secret[7] * secret[2] == 1179995)
s.add(secret[15] * (secret[12] + 1) + secret[14] == 11345)
s.add(secret[19] * secret[18] - secret[20] * secret[4] - secret[13] == -326)
s.add(secret[3] * secret[0] * secret[5] == 1597956)
s.add(secret[3] * secret[9] - secret[8] == 12200)
s.add(secret[1] - secret[5] * secret[9] - secret[5] + secret[1] == -13114)
s.add(secret[11] * secret[4] + secret[9] == 12423)
s.add(secret[14] * secret[19] + secret[3] == 12654)
s.add(secret[16] * secret[0] * secret[4] * secret[18] == 134197560)
s.add(secret[17] + secret[16] * secret[19] + secret[13] * secret[7] ==
20478)
s.add(secret[14] + secret[4] * secret[7] - secret[8] == 10252)
s.add(secret[17] + secret[0] * secret[10] * secret[11] == 1627352)
s.add(secret[17] + secret[16] - secret[15] + secret[12] == 191)
s.add(secret[8] + secret[5] * secret[14] == 13455)
s.add(secret[5] * secret[2] == 13570)
```

```
s.add(secret[20] - secret[8] + secret[1] * secret[12] - secret[12] == 4739)
s.add(secret[5] + secret[6] + secret[9] == 330)

print(s.check())
model = s.model()
result = ''.join([chr(int(str(model[secret[i]]))) for i in
range(len(model))])
print(result)
assert md5(result.encode()).hexdigest() ==
'bfe0f7cd0a926ec05cee3717bd9bce20'
```

solve.py

```
import os
import subprocess
from pwn import xor

secret = b"z3solve_your_equation"
secret_int = int.from_bytes(secret, byteorder='big')
for i in range(4):
    secret_int >>= secret[i * 4]
    secret_int <<= secret[i * 4]

secret = secret_int.to_bytes(len(secret), 'big')

path = "ransom/"
listOfFiles = subprocess.check_output(f'ls {path} | grep .pdf',
shell=True).split(b'\n')[:-1]

signature = bytes.fromhex("255044462D")
for file in listOfFiles:
    content = open(f"{path}{file.decode()}","rb").read()[1337:-1337]
    secret2 = xor(content[:25], signature)[:len(secret)]
    rsecret = xor(secret, secret2)

    newContent = signature
    newContent += xor(content[25:], rsecret)
    open(f"decrypted_{file.decode()}","wb+").write(newContent)
```

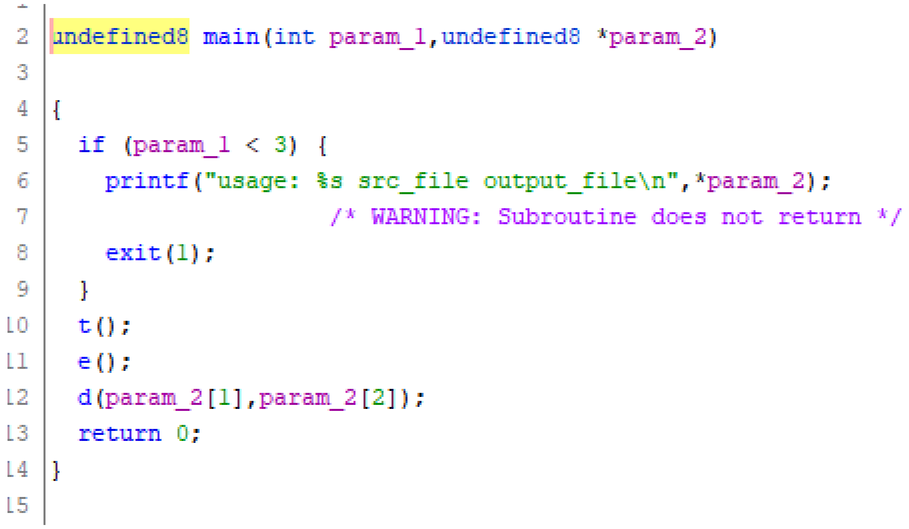
Flag:

CTFTED2021{recover_likely_ransomfiles_with_simple_z3}

Rotat-eat

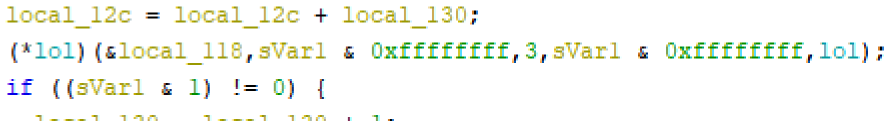
Langkah Penyelesaian:

Langung decompile file ELFnya



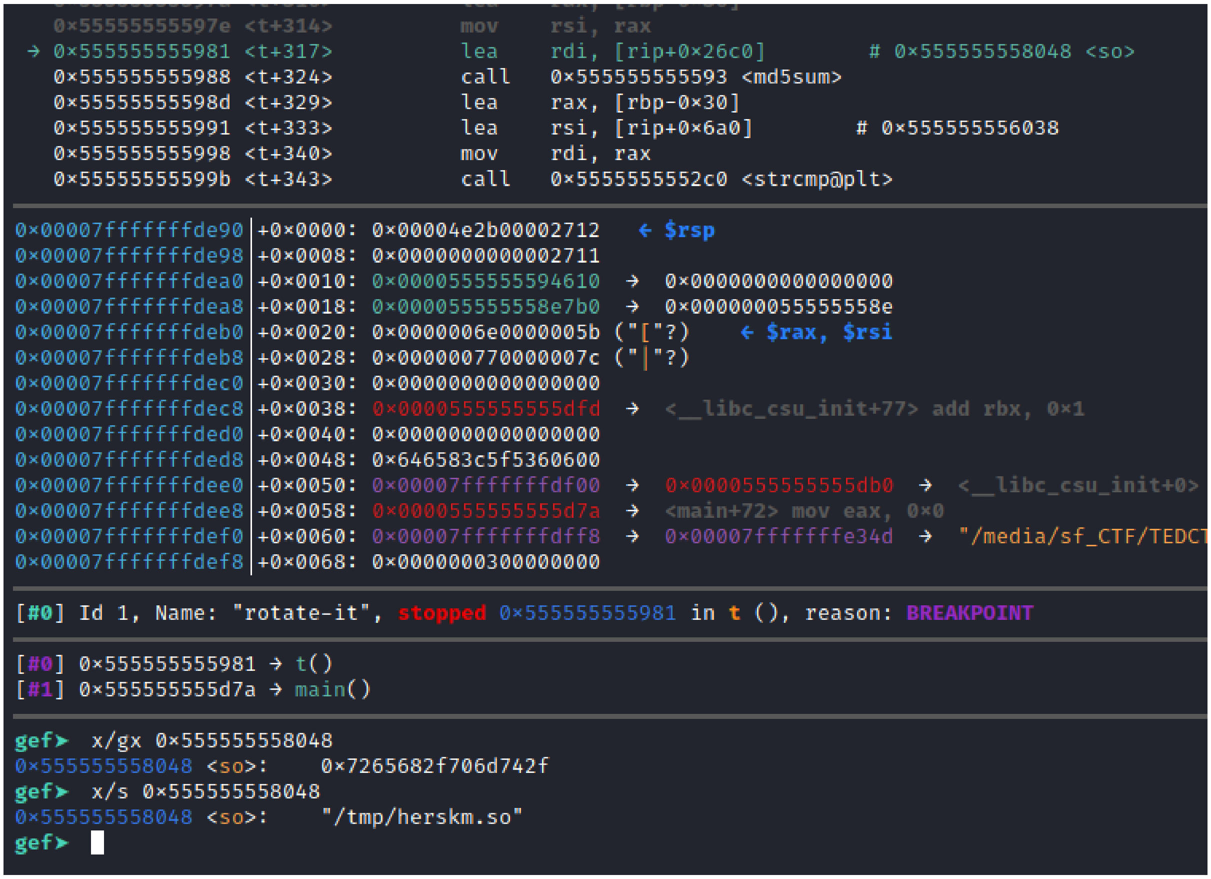
```
undefined8 main(int param_1,undefined8 *param_2)

{
  if (param_1 < 3) {
    printf("usage: %s src_file output_file\n",*param_2);
                    /* WARNING: Subroutine does not return */
    exit(1);
  }
  t();
  e();
  d(param_2[1],param_2[2]);
  return 0;
}

```

Function t dan e untuk mengambil file tapi tidak tau file apa, dan dimasukan ke variable lol.
Function d untuk decrypt src file, dan dimasukan ke output file, decryptnya memanggil variable lol.



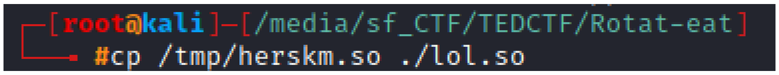
```
  local_12c = local_12c + local_130;
  (*lol)(&local_118,sVar1 & 0xffffffff,3,sVar1 & 0xffffffff,lol);
  if ((sVar1 & 1) != 0) {
```

Pertama kali kepikiran adalah mengambil file library yang didalam ada function decrypt text.
Untuk mencari file dimana ditaruh, Saya pakai gdb untuk mencari tau dimana file nya di write.



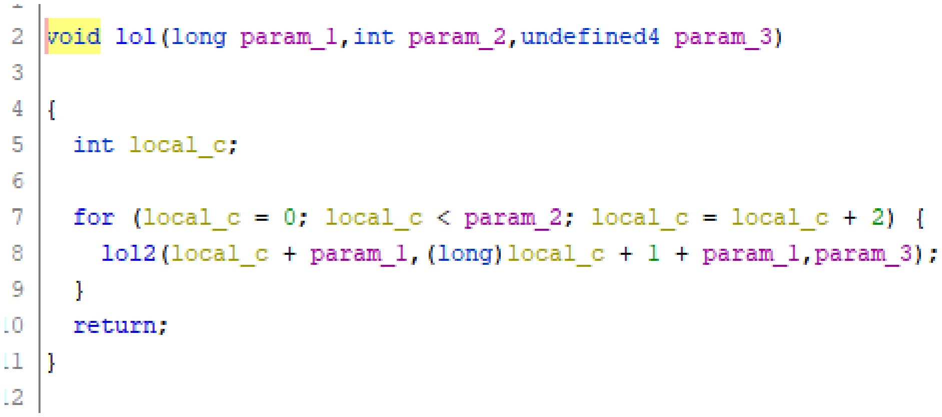
```
   0x55555555597e <t+314>          mov    rsi, rax
 → 0x555555555981 <t+317>          lea    rdi, [rip+0x26c0]        # 0x555555558048 <so>
   0x555555555988 <t+324>          call   0x555555555593 <md5sum>
   0x55555555598d <t+329>          lea    rax, [rbp-0x30]
   0x555555555991 <t+333>          lea    rsi, [rip+0x6a0]        # 0x555555556038
   0x555555555998 <t+340>          mov    rdi, rax
   0x55555555599b <t+343>          call   0x5555555552c0 <strcmp@plt>

0x00007fffffffde90│+0x0000: 0x00004e2b00002712   ← $rsp
0x00007fffffffde98│+0x0008: 0x0000000000002711
0x00007fffffffdea0│+0x0010: 0x0000555555594610  →  0x0000000000000000
0x00007fffffffdea8│+0x0018: 0x00005555558e7b0  →  0x000000055555558e
0x00007fffffffdeb0│+0x0020: 0x0000006e0000005b ("["?)    ← $rax, $rsi
0x00007fffffffdeb8│+0x0028: 0x000000770000007c ("|"?)
0x00007fffffffdec0│+0x0030: 0x0000000000000000
0x00007fffffffdec8│+0x0038: 0x0000555555555dfd  →  <__libc_csu_init+77> add rbx, 0x1
0x00007fffffffded0│+0x0040: 0x0000000000000000
0x00007fffffffded8│+0x0048: 0x646583c5f5360600
0x00007fffffffdee0│+0x0050: 0x00007fffffffdf00  →  0x0000555555555db0  →  <__libc_csu_init+0>
0x00007fffffffdee8│+0x0058: 0x0000555555555d7a  →  <main+72> mov eax, 0x0
0x00007fffffffdef0│+0x0060: 0x00007fffffffdff8  →  0x00007fffffffe34d  →  "/media/sf_CTF/TEDCT
0x00007fffffffdef8│+0x0068: 0x0000000300000000

[#0] Id 1, Name: "rotate-it", stopped 0x555555555981 in t (), reason: BREAKPOINT

[#0] 0x555555555981 → t()
[#1] 0x555555555d7a → main()

gef➤  x/gx 0x555555558048
0x555555558048 <so>:    0x7265682f706d742f
gef➤  x/s 0x555555558048
0x555555558048 <so>:    "/tmp/herskm.so"
gef➤  
```

Copy filenya



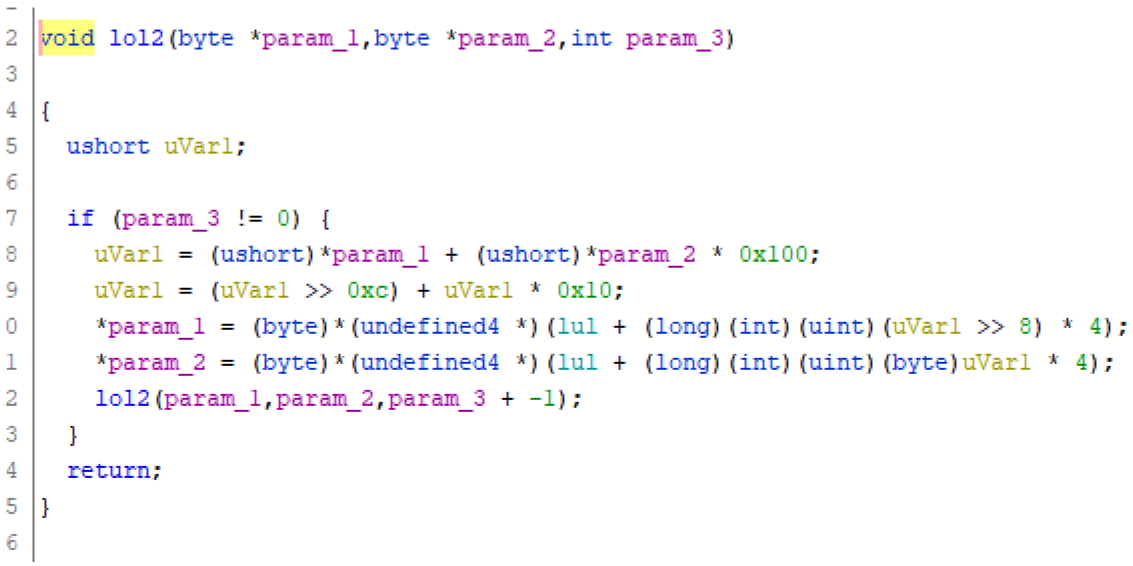
```
┌─[root@kali]─[/media/sf_CTF/TEDCTF/Rotat-eat]
└──╼ #cp /tmp/herskm.so ./lol.so
```

Decompiler untuk function lol



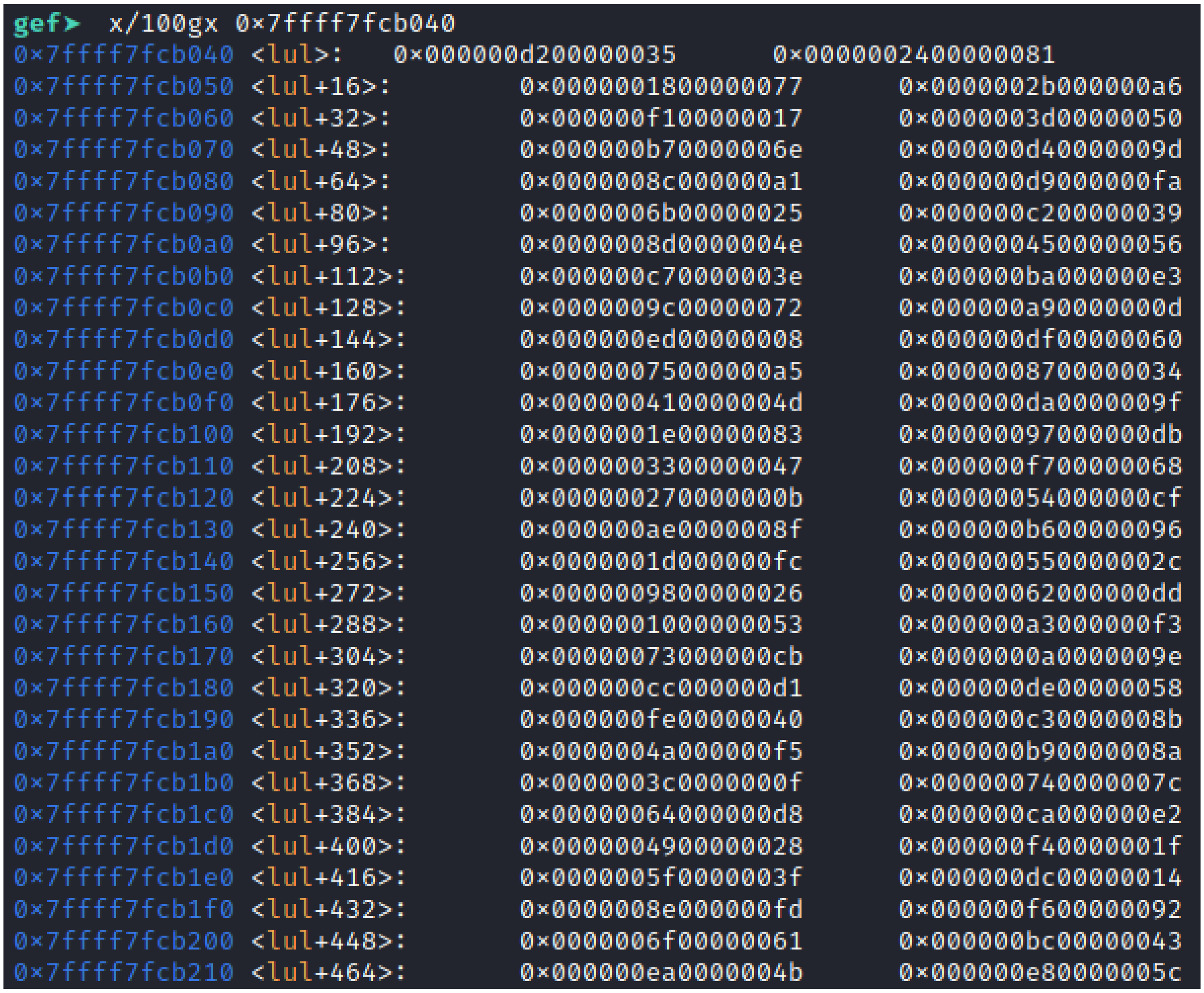
```

void lol(long param_1,int param_2,undefined4 param_3)

{
  int local_c;
  
  for (local_c = 0; local_c < param_2; local_c = local_c + 2) {
    lol2(local_c + param_1,(long)local_c + 1 + param_1,param_3);
  }
  return;
}

```

Decompiler untuk function lol2



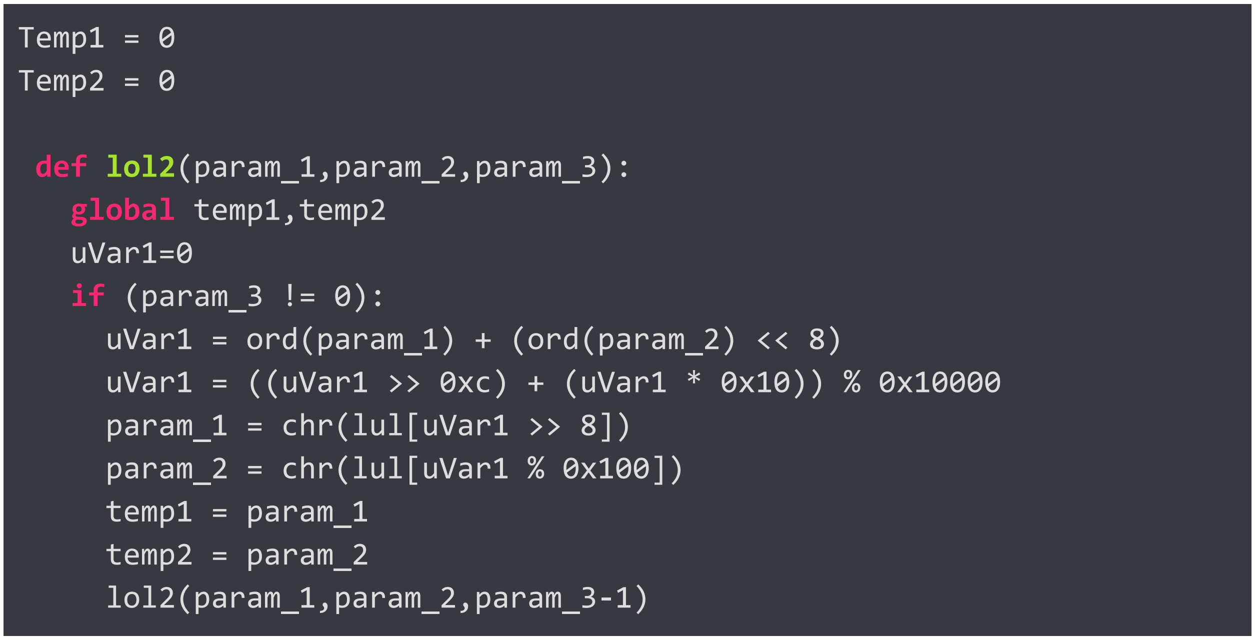
```
void lol2(byte *param_1,byte *param_2,int param_3)

{
  ushort uVar1;

  if (param_3 != 0) {
    uVar1 = (ushort)*param_1 + (ushort)*param_2 * 0x100;
    uVar1 = (uVar1 >> 0xc) + uVar1 * 0x10;
    *param_1 = (byte)*(undefined4 *)(lul + (long)(int)(uint)(uVar1 >> 8) * 4);
    *param_2 = (byte)*(undefined4 *)(lul + (long)(int)(uint)(byte)uVar1 * 4);
    lol2(param_1,param_2,param_3 + -1);
  }
  return;
}
```

Yang paling penting adalah function lol2, didalam lol2 ada array int lul, saya cari valuenya menggunakan gdb dan copy secara manual



```
gef➤  x/100gx 0x7ffff7fcb040
0x7ffff7fcb040 <lul>:   0x000000d200000035      0x0000002400000081
0x7ffff7fcb050 <lul+16>:        0x0000001800000077      0x0000002b000000a6
0x7ffff7fcb060 <lul+32>:        0x000000f100000017      0x0000003d00000050
0x7ffff7fcb070 <lul+48>:        0x000000b70000006e      0x000000d40000009d
0x7ffff7fcb080 <lul+64>:        0x0000008c000000a1      0x000000d9000000fa
0x7ffff7fcb090 <lul+80>:        0x0000006b00000025      0x000000c200000039
0x7ffff7fcb0a0 <lul+96>:        0x0000008d0000004e      0x0000004500000056
0x7ffff7fcb0b0 <lul+112>:       0x000000c70000003e      0x000000ba000000e3
0x7ffff7fcb0c0 <lul+128>:       0x0000009c00000072      0x000000a90000000d
0x7ffff7fcb0d0 <lul+144>:       0x000000ed00000008      0x000000df00000060
0x7ffff7fcb0e0 <lul+160>:       0x00000075000000a5      0x0000008700000034
0x7ffff7fcb0f0 <lul+176>:       0x000000410000004d      0x000000da0000009f
0x7ffff7fcb100 <lul+192>:       0x0000001e00000083      0x00000097000000db
0x7ffff7fcb110 <lul+208>:       0x0000003300000047      0x000000f700000068
0x7ffff7fcb120 <lul+224>:       0x000000270000000b      0x00000054000000cf
0x7ffff7fcb130 <lul+240>:       0x000000ae0000008f      0x000000b600000096
0x7ffff7fcb140 <lul+256>:       0x0000001d000000fc      0x000000550000002c
0x7ffff7fcb150 <lul+272>:       0x0000009800000026      0x00000062000000dd
0x7ffff7fcb160 <lul+288>:       0x0000001000000053      0x000000a3000000f3
0x7ffff7fcb170 <lul+304>:       0x00000073000000cb      0x0000000a0000009e
0x7ffff7fcb180 <lul+320>:       0x000000cc000000d1      0x000000de00000058
0x7ffff7fcb190 <lul+336>:       0x000000fe00000040      0x000000c30000008b
0x7ffff7fcb1a0 <lul+352>:       0x0000004a000000f5      0x000000b90000008a
0x7ffff7fcb1b0 <lul+368>:       0x0000003c0000000f      0x000000740000007c
0x7ffff7fcb1c0 <lul+384>:       0x00000064000000d8      0x000000ca000000e2
0x7ffff7fcb1d0 <lul+400>:       0x0000004900000028      0x000000f40000001f
0x7ffff7fcb1e0 <lul+416>:       0x0000005f0000003f      0x000000dc00000014
0x7ffff7fcb1f0 <lul+432>:       0x0000008e000000fd      0x000000f600000092
0x7ffff7fcb200 <lul+448>:       0x0000006f00000061      0x000000bc00000043
0x7ffff7fcb210 <lul+464>:       0x000000ea0000004b      0x000000e80000005c
```

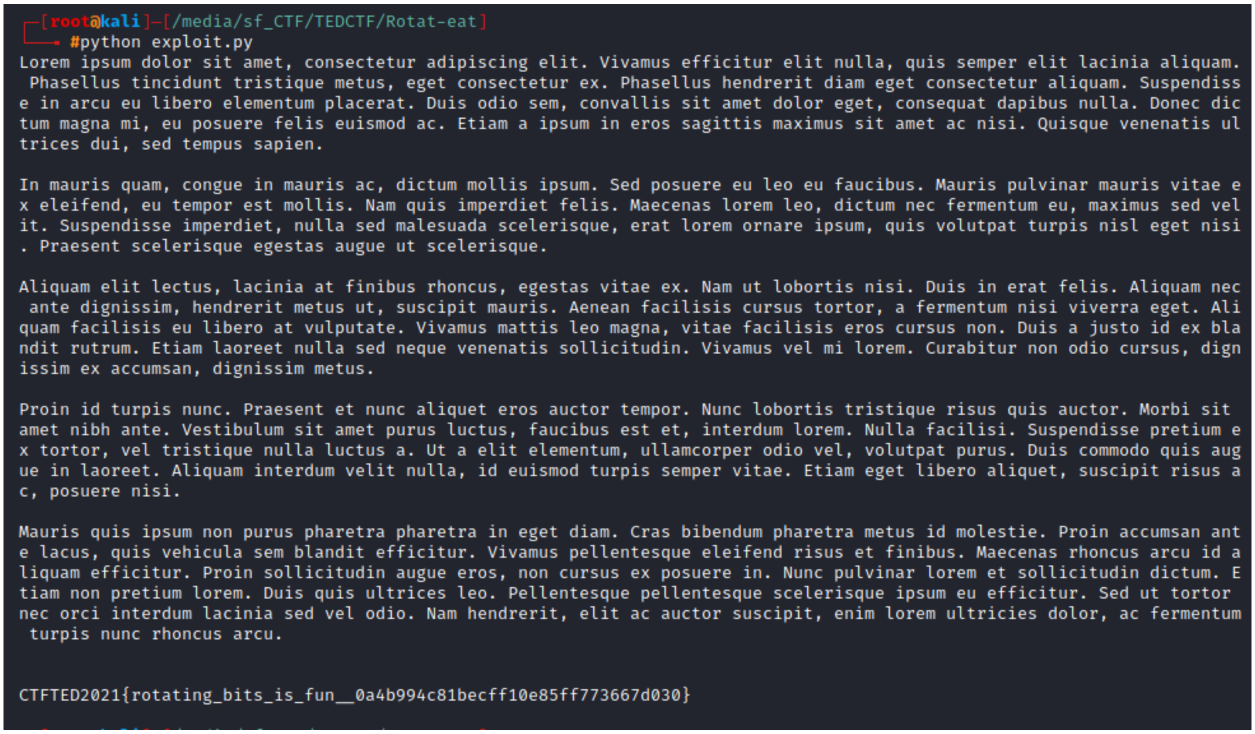
setelah disesuaikan dengan languange python, function lol2 akan begini



```
Temp1 = 0
Temp2 = 0

 def lol2(param_1,param_2,param_3):
   global temp1,temp2
   uVar1=0
   if (param_3 != 0):
     uVar1 = ord(param_1) + (ord(param_2) << 8)
     uVar1 = ((uVar1 >> 0xc) + (uVar1 * 0x10)) % 0x10000
     param_1 = chr(lul[uVar1 >> 8])
     param_2 = chr(lul[uVar1 % 0x100])
     temp1 = param_1
     temp2 = param_2
     lol2(param_1,param_2,param_3-1)
```

Inti alurnya mengambil 2 byte setiap looping dan decrypt dengan recursive 3 kali di function lol2.

Tinggal reverse function lol2, pada decrypt text bisa mendapatkan uVar1 tetapi untuk mendapatkan uVar1 selanjutnya sulit, jadi saya menggunakan brute force 0-255 di param_1 dan param_2, setelah itu tinggal dicocokan dengan uVar1 dari decrypt text, terus looping 3 kali.



```
┌─[root@kali]─[/media/sf_CTF/TEDCTF/Rotat-eat]
└──╼ #python exploit.py
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus efficitur elit nulla, quis semper elit lacinia aliquam.
 Phasellus tincidunt tristique metus, eget consectetur ex. Phasellus hendrerit diam eget consectetur aliquam. Suspendiss
e in arcu eu libero elementum placerat. Duis odio sem, convallis sit amet dolor eget, consequat dapibus nulla. Donec dic
tum magna mi, eu posuere felis euismod ac. Etiam a ipsum in eros sagittis maximus sit amet ac nisi. Quisque venenatis ul
trices dui, sed tempus sapien.

In mauris quam, congue in mauris ac, dictum mollis ipsum. Sed posuere eu leo eu faucibus. Mauris pulvinar mauris vitae e
x eleifend, eu tempor est mollis. Nam quis imperdiet felis. Maecenas lorem leo, dictum nec fermentum eu, maximus sed vel
it. Suspendisse imperdiet, nulla sed malesuada scelerisque, erat lorem ornare ipsum, quis volutpat turpis nisl eget nisi
. Praesent scelerisque egestas augue ut scelerisque.

Aliquam elit lectus, lacinia at finibus rhoncus, egestas vitae ex. Nam ut lobortis nisi. Duis in erat felis. Aliquam nec
 ante dignissim, hendrerit metus ut, suscipit mauris. Aenean facilisis cursus tortor, a fermentum nisi viverra eget. Ali
quam facilisis eu libero at vulputate. Vivamus mattis leo magna, vitae facilisis eros cursus non. Duis a justo id ex bla
ndit rutrum. Etiam laoreet nulla sed neque venenatis sollicitudin. Vivamus vel mi lorem. Curabitur non odio cursus, dign
issim ex accumsan, dignissim metus.

Proin id turpis nunc. Praesent et nunc aliquet eros auctor tempor. Nunc lobortis tristique risus quis auctor. Morbi sit
amet nibh ante. Vestibulum sit amet purus luctus, faucibus est et, interdum lorem. Nulla facilisi. Suspendisse pretium e
x tortor, vel tristique nulla luctus a. Ut a elit elementum, ullamcorper odio vel, volutpat purus. Duis commodo quis aug
ue in laoreet. Aliquam interdum velit nulla, id euismod turpis semper vitae. Etiam eget libero aliquet, suscipit risus a
c, posuere nisi.

Mauris quis ipsum non purus pharetra pharetra in eget diam. Cras bibendum pharetra metus id molestie. Proin accumsan ant
e lacus, quis vehicula sem blandit efficitur. Vivamus pellentesque eleifend risus et finibus. Maecenas rhoncus arcu id a
liquam efficitur. Proin sollicitudin augue eros, non cursus ex posuere in. Nunc pulvinar lorem et sollicitudin dictum. E
tiam non pretium lorem. Duis quis ultrices leo. Pellentesque pellentesque scelerisque ipsum eu efficitur. Sed ut tortor
nec orci interdum lacinia sed vel odio. Nam hendrerit, elit ac auctor suscipit, enim lorem ultricies dolor, ac fermentum
 turpis nunc rhoncus arcu.


CTFTED2021{rotating_bits_is_fun__0a4b994c81becff10e85ff773667d030}
```

Code:

exploit.py

```
lul =
[0x35,0xd2,f,00x81,0x24,0x77,0x18,0xa6,0x2b,0x17,0xf1,0x50,0x3d,0x
6e,0xb7,0x9d,0xd4,0xa1,0x8c,0xfa,0xd9,0x25,0x6b,0x39,0xc2,0x4e,0x8
d,0x56,0x45,0x3e,0xc7,0xe3,0xba,0x72,0x9c,0x0d,0xa9,0x08,0xed,0x60
,0xdxa5,0x75,0x34,0x87,0x4d,0x41,0x9f,0xda,0x83,0x1e,0xdb,0x97,0x4
7,0x33,0x68,0xf7,0x0b,0x27,0xcf,0x54,0x8f,0xae,0x96,0xb6,0xfc,0x1d
,0x2c,0x55,0x26,0x98,0xdd,0x62,0x53,0x10,0xf3,0xa3,0xcb,0x73,0x9e,
0x0a,0xd1,0xcc,0x58,0xde,0x40,0xfe,0x8b,0xc3,0xf5,0x4a,0x8a,0xb9,0
x0f,0x3c,0x7c,0x74,0xd8,0x64,0xe2,0xca,0x28,0x49,0x1f,0xf4,0x3f,0x
5f,0x14,0xdc,0xfd,0x8e,0x92,0xf6,0x61,0x6f,0x43,0xbc,0x4b,0xea,0x5
c,0xe8,0x88,0xe6,0x70,0x36,0xa8,0xb5,0x89,0x4f,0xc9,0x95,0x32,0x1b
,0x2a,0x66,0x11,0xbf,0xb2,0x04,0x3a,0x71,0xc1,0xd0,0x9a,0x99,0xff,
0x07,0xb0,0x2f,0xbe,0xac,0xaa,0x67,0x7d,0x1a,0xa7,0x65,0x9b,0xe0,0
xc8,0xf0,0x7a,0x44,0x7b,0x0c,0x86,0x91,0x03,0xf2,0x69,0xd7,0xb3,0x
2d,0xe4,0x7e,0x5d,0xd6,0x48,0x22,0x01,0x16,0x05,0x57,0x37,0xfb,0x1
9,0xef,0x6d,0x51,0xa0,0xe9,0xad,0xcd,0xab,0xc4,0x52,0x0e,0x7f,0x85
,0xeb,0x90,0xd3,0xc5,0xc6,0x78,0x5e,0xce,0x21,0x20,0xb4,0xe1,0xe5,
0x3b,0x46,0x13,0xb1,0x30,0x80,0xa4,0xf8,0xbb,0x93,0x29,0x79,0xb8,0
xbd,0x1c,0xc0,0x09,0x59,0x63,0x06,0xec,0xee,0x82,0x84,0x5b,0x6c,0x
6a,0x23,0x31,0x76,0x4c,0x2e,0xd5,0x94,0xa2,0x00,0x5a,0x02,0x12,0xf
9,0xaf,0x15,0xe7,0x38,0x42]

with open("secret.txt.enc","rb") as f:
  dat = f.read()

def reverse_lol2(idx1):
  for param_1 in range(255+1):
    for param_2 in range(255+1):
      uVar1 = param_1 + (param_2 << 8)
      uVar1 = ((uVar1 >> 0xc) + (uVar1 * 0x10)) % 0x10000

      if (idx1==uVar1) :
        return chr(param_1),chr(param_2)

final = []
for i in range(0,len(dat)-5,2):
  temp1 = dat[i]
  temp2 = dat[i+1]
  for _ in range(3):
    idx1 = lul.index(ord(temp1))
    idx2 = lul.index(ord(temp2))
    idx1 = (idx1 << 8) + idx2
    temp1,temp2 = reverse_lol2(idx1)
  final.append(temp1)
  final.append(temp2)

print "".join(final)
```

Flag:

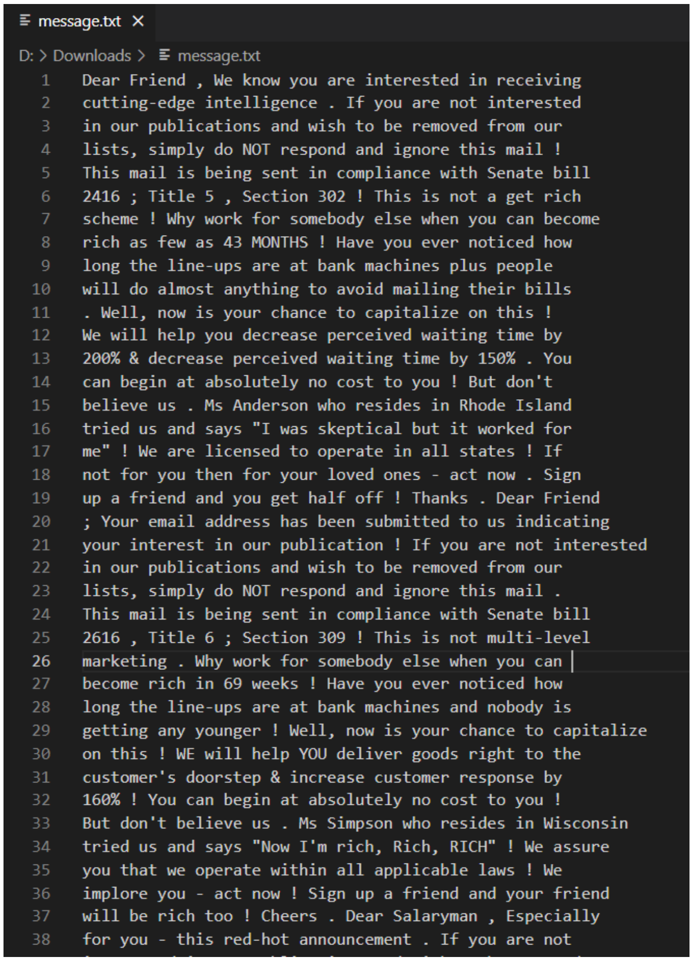
CTFTED2021{rotating_bits_is_fun__0a4b994c81becff10e85ff773667d030}

Steganography

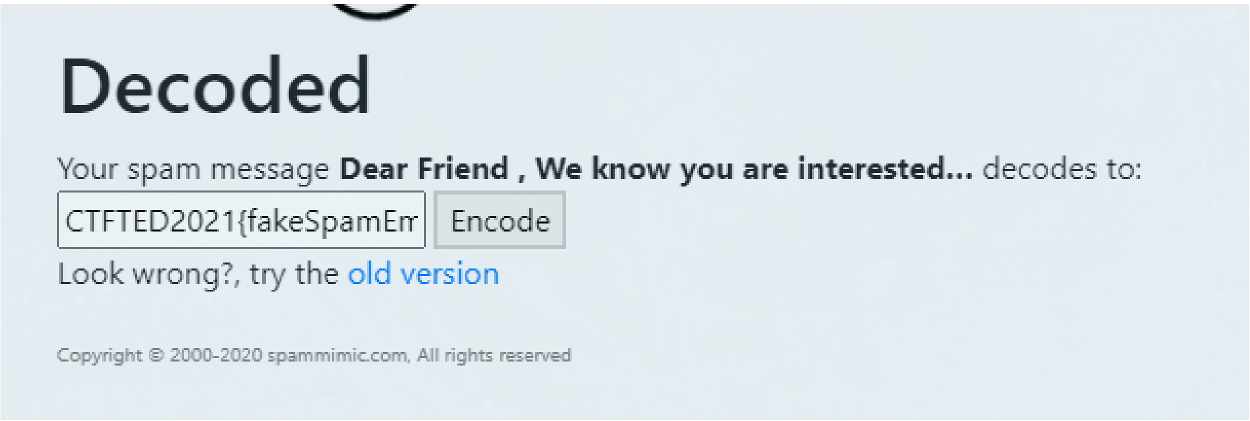
Dear Friend

Langkah Penyelesaian:

Diberikan sebuah file message.txt yang berisi seperti email spam atau penipuan, mengingat ini challenge Steganography langsung teringat tool online bernama spamimic.



```
Dear Friend , We know you are interested in receiving
cutting-edge intelligence . If you are not interested
in our publications and wish to be removed from our
lists, simply do NOT respond and ignore this mail !
This mail is being sent in compliance with Senate bill
2416 ; Title 5 , Section 302 ! This is not a get rich
scheme ! Why work for somebody else when you can become
rich as few as 43 MONTHS ! Have you ever noticed how
long the line-ups are at bank machines plus people
will do almost anything to avoid mailing their bills
. Well, now is your chance to capitalize on this !
We will help you decrease perceived waiting time by
200% & decrease perceived waiting time by 150% . You
can begin at absolutely no cost to you ! But don't
believe us . Ms Anderson who resides in Rhode Island
tried us and says "I was skeptical but it worked for
me" ! We are licensed to operate in all states ! If
not for you then for your loved ones - act now . Sign
up a friend and you get half off ! Thanks . Dear Friend
; Your email address has been submitted to us indicating
your interest in our publication ! If you are not interested
in our publications and wish to be removed from our
lists, simply do NOT respond and ignore this mail .
This mail is being sent in compliance with Senate bill
2616 , Title 6 ; Section 309 ! This is not multi-level
marketing . Why work for somebody else when you can
become rich in 69 weeks ! Have you ever noticed how
long the line-ups are at bank machines and nobody is
getting any younger ! Well, now is your chance to capitalize
on this ! WE will help YOU deliver goods right to the
customer's doorstep & increase customer response by
160% ! You can begin at absolutely no cost to you !
But don't believe us . Ms Simpson who resides in Wisconsin
tried us and says "Now I'm rich, Rich, RICH" ! We assure
you that we operate within all applicable laws ! We
implore you - act now ! Sign up a friend and your friend
will be rich too ! Cheers . Dear Salaryman , Especially
for you - this red-hot announcement . If you are not
```



Decoded

Your spam message **Dear Friend , We know you are interested...** decodes to:

CTFTED2021{fakeSpamEm

Look wrong?, try the old version

Copyright © 2000-2020 spammimic.com, All rights reserved

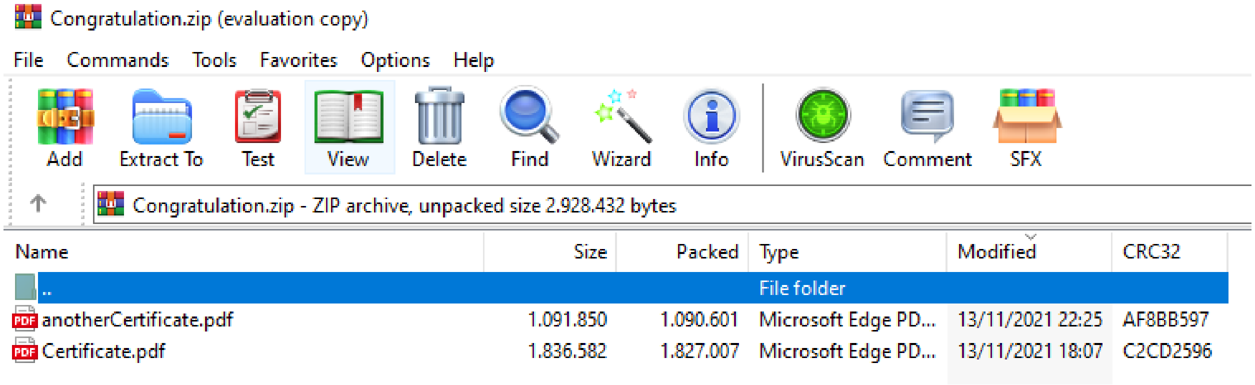
Flag:

`CTFTED2021{fakeSpamEmail_turn_out_to_be_important_message}`

Congratulations

Langkah Penyelesaian:

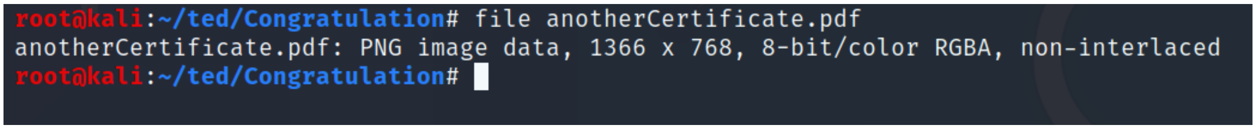
Diberikan 2 file PDF



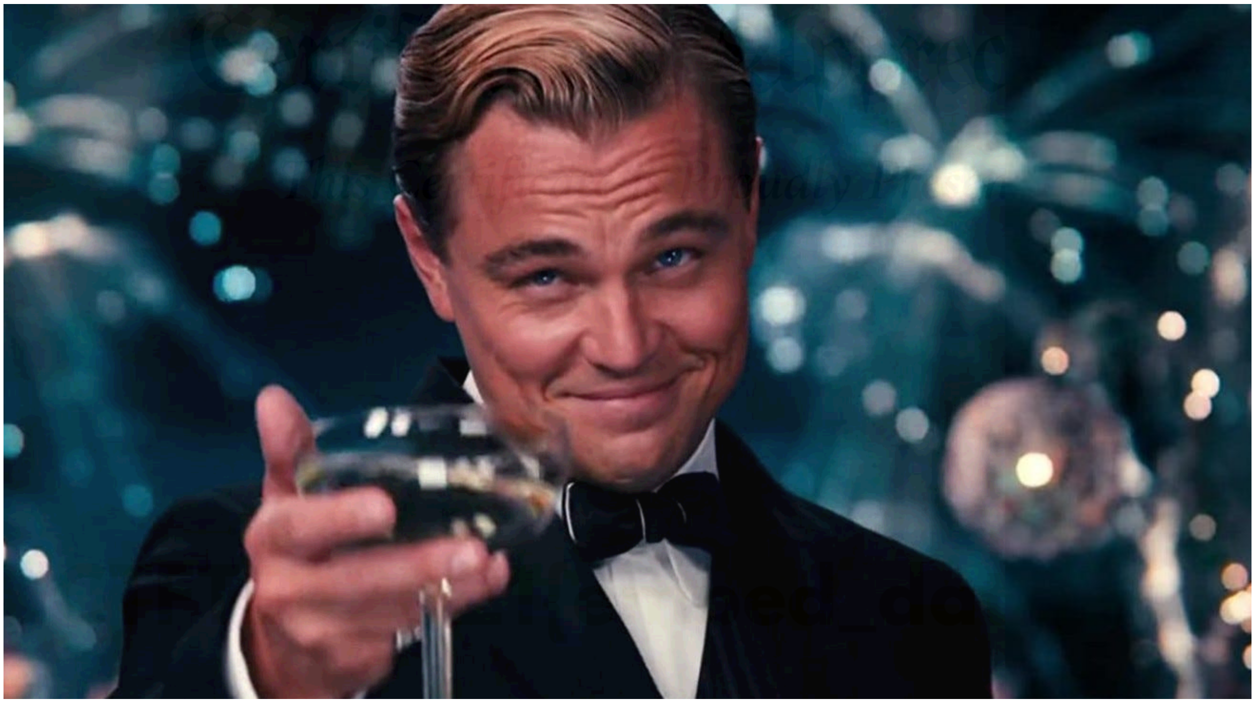
Congratulation.zip - ZIP archive, unpacked size 2.928.432 bytes

Name	Size	Packed	Type	Modified	CRC32
..			File folder		
anotherCertificate.pdf	1.091.850	1.090.601	Microsoft Edge PD...	13/11/2021 22:25	AF8BB597
Certificate.pdf	1.836.582	1.827.007	Microsoft Edge PD...	13/11/2021 18:07	C2CD2596

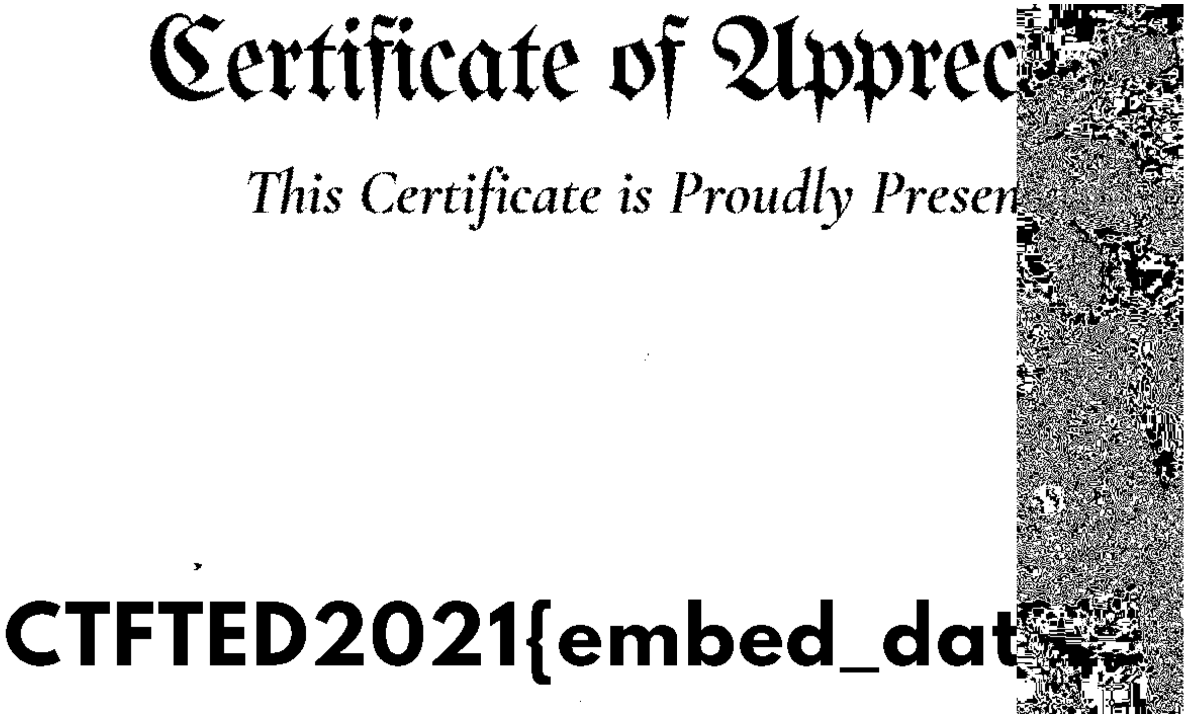
File anotherCertificate ternyata rusak dan setelah di cek sepertinya berupa PNG, jadi tinggal ganti extension file.



```
root@kali:~/ted/Congratulation# file anotherCertificate.pdf
anotherCertificate.pdf: PNG image data, 1366 x 768, 8-bit/color RGBA, non-interlaced
root@kali:~/ted/Congratulation# 
```



Di belakang jidat diCaprio seperti ada tulisan, mari kita stegsolve / aperisolve



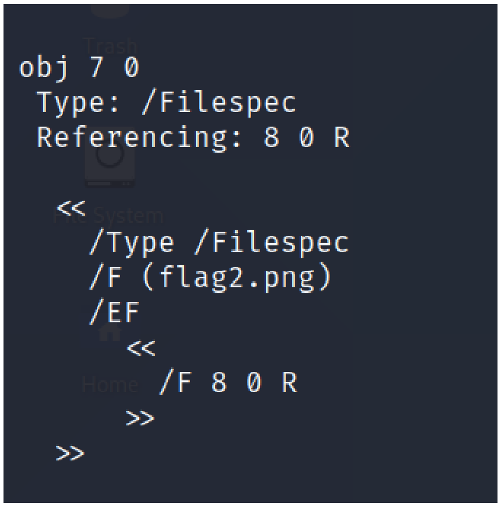
Certificate of Apprec

This Certificate is Proudly Presen

CTFTED2021{embed_dat

Didapatkan setengah flag

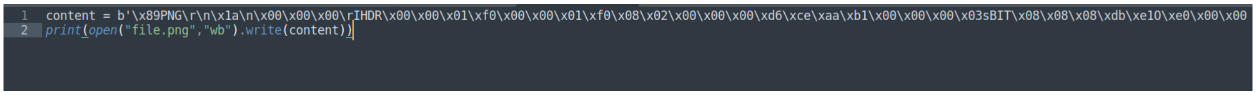
Ternyata di file yang sama terdapat



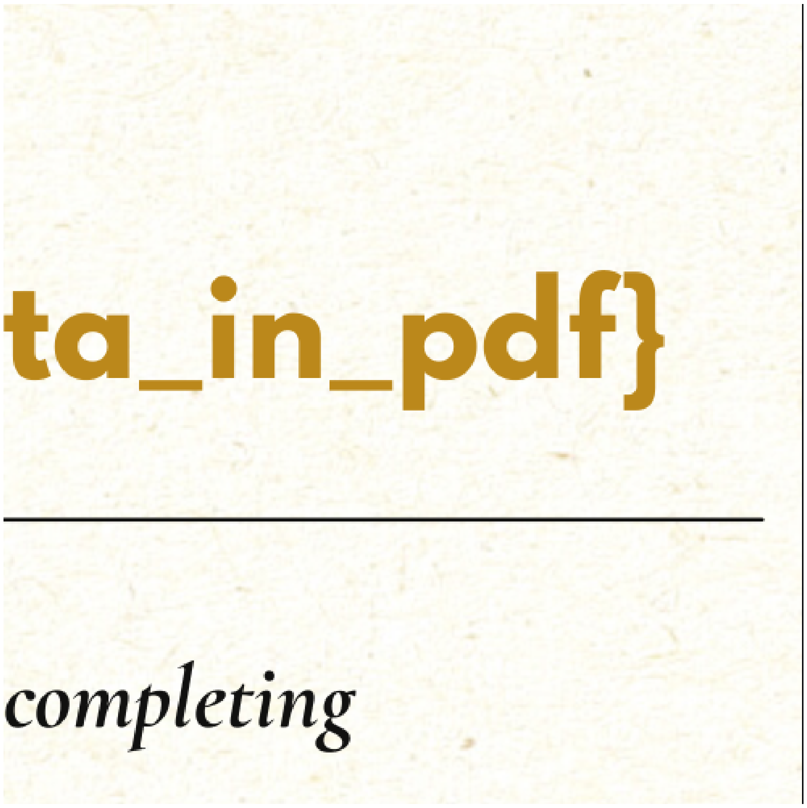
```
obj 7 0
 Type: /Filespec
 Referencing: 8 0 R

  <<
    /Type /Filespec
    /F (flag2.png)
    /EF
      <<
        /F 8 0 R
      >>
  >>
```

```
pdf-parser --stats anotherCertificate.pdf
pdf-parser --object 8 --raw --filter anotherCertificate.pdf >
out
```

Lalu dijadikan png menggunakan python



```
content = b'\x89PNG\r\n\x1a\n\x00\x00\x00\rIHDR\x00\x00\x01\xf0\x00\x00\x01\xf0\x08\x02\x00\x00\x00\xd6\xce\xaa\xb1\x00\x00\x00\x03sBIT\x08\x08\x08\xdb\xe1O\xe0\x00\x00
print(open("file.png","wb").write(content))
```



ta_in_pdf}

completing

Flag:

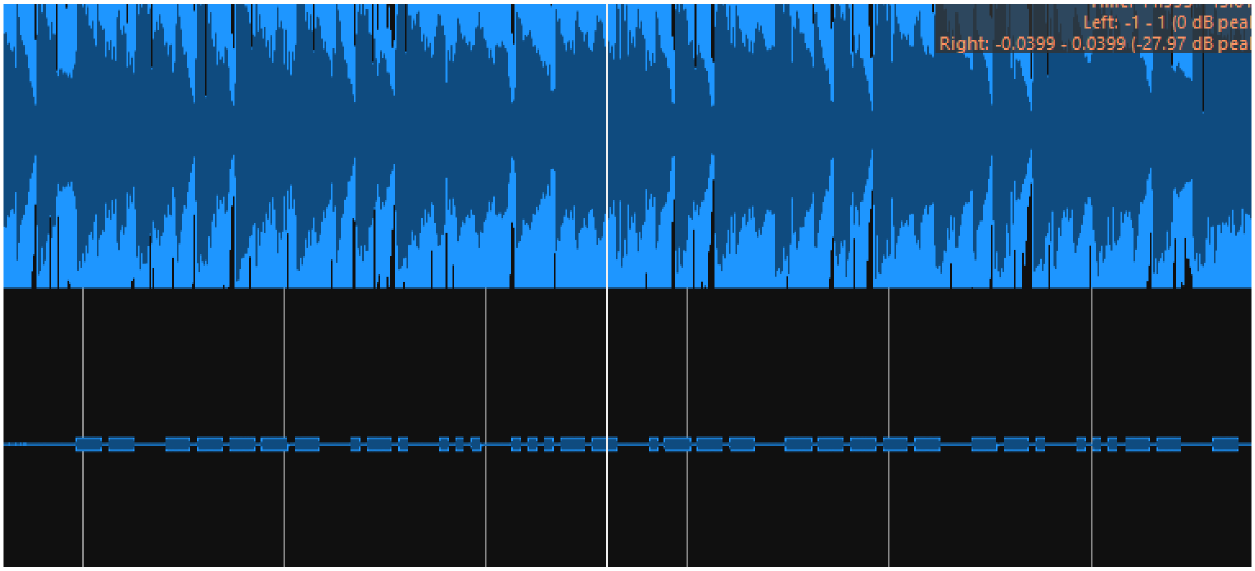
`CTFTED2021{embed_data_in_pdf}`

Forensic

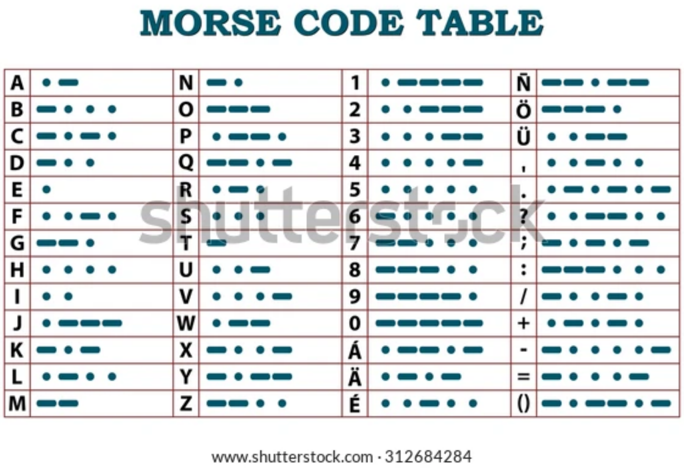
Dance

Langkah Penyelesaian:

Diberikan audio file karena saya cupu saya masukin sonic visualizer liat spectogramnya



Ternyata ada 2 channel, channel yg dibawah kinda sus mirip morse code, jadi dicoba saja untuk di decode menggunakan tabel.



MORSE CODE TABLE

A	•—	N	—•	1	•————	Ñ	——•——
B	—•••	O	———	2	••———	Ö	———•
C	—•—•	P	•——•	3	•••——	Ü	••——
D	—••	Q	——•—	4	••••—	,	••—••
E	•	R	•—•	5	•••••	.	•—•—•—
F	••—•	S	•••	6	—••••	?	••——••
G	——•	T	—	7	——•••	;	—•—•—
H	••••	U	••—	8	———••	:	———•••
I	••	V	•••—	9	————•	/	—••—•
J	•———	W	•——	0	—————	+	•—•—•
K	—•—	X	—••—	Á	•——•—	-	—••••—
L	•—••	Y	—•——	Ä	•—•—	=	—•••—
M	——	Z	——••	É	••—••	()	—•——•—

www.shutterstock.com · 312684284

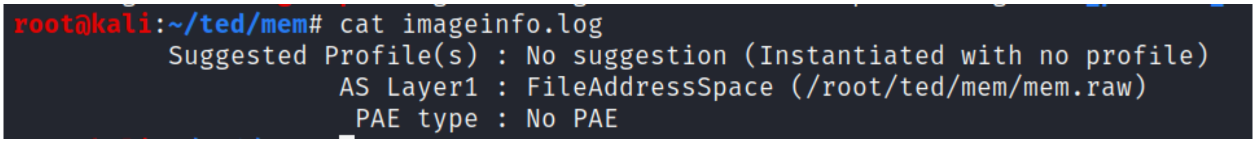
Flag:

CTFTED2021{M0RS3J0G3T}

crash

Langkah Penyelesaian:

Sepertinya challenge volatile memory, pertama di imageinfo tapi tidak menemukan profile yang cocok



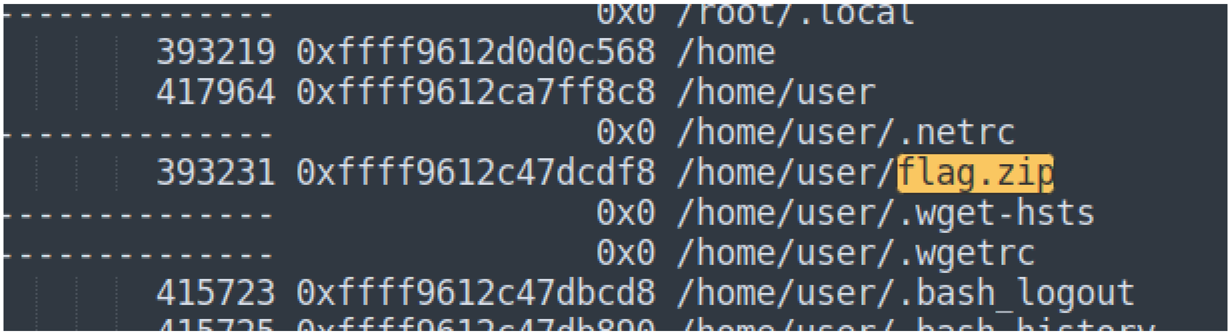
```
root@kali:~/ted/mem# cat imageinfo.log
          Suggested Profile(s) : No suggestion (Instantiated with no profile)
                     AS Layer1 : FileAddressSpace (/root/ted/mem/mem.raw)
                      PAE type : No PAE
```

Kemungkinan besar linux profile, jadi menggunakan string magic mendapatkan distro dan boot image nya

Ubuntu20.04-5.4.0-42-generic

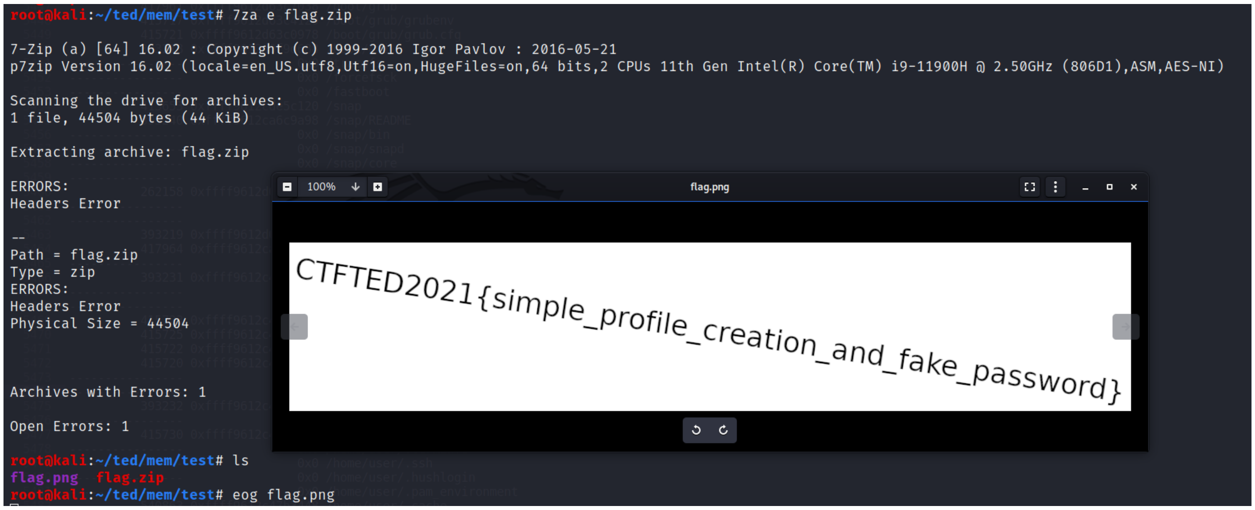
Pembuatan image bisa menggunakan docker / vm

```
python /opt/volatility/vol.py -f mem.raw
--profile=LinuxUbuntu20_04-5_4_0-42-genericx64 linux_find_file
-L | tee list_file.log
```



```
----------------                0x0 /root/.local
       393219 0xffff9612d0d0c568 /home
       417964 0xffff9612ca7ff8c8 /home/user
----------------                0x0 /home/user/.netrc
       393231 0xffff9612c47dcdf8 /home/user/flag.zip
----------------                0x0 /home/user/.wget-hsts
----------------                0x0 /home/user/.wgetrc
       415723 0xffff9612c47dbcd8 /home/user/.bash_logout
       415725 0xffff9612c47db890 /home/user/.bash_history
```

```
python /opt/volatility/vol.py -f mem.raw
--profile=LinuxUbuntu20_04-5_4_0-42-genericx64 linux_find_file
-i 0xffff9612c47dcdf8 -O ./flag.zip
```



```
root@kali:~/ted/mem/test# 7za e flag.zip

7-Zip (a) [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.utf8,Utf16=on,HugeFiles=on,64 bits,2 CPUs 11th Gen Intel(R) Core(TM) i9-11900H @ 2.50GHz (806D1),ASM,AES-NI)

Scanning the drive for archives:
1 file, 44504 bytes (44 KiB)

Extracting archive: flag.zip

ERRORS:
Headers Error

--
Path = flag.zip
Type = zip
ERRORS:
Headers Error
Physical Size = 44504


Archives with Errors: 1

Open Errors: 1

root@kali:~/ted/mem/test# ls
flag.png  flag.zip
root@kali:~/ted/mem/test# eog flag.png
```

CTFTED2021{simple_profile_creation_and_fake_password}

Didapatkan flag

Flag:

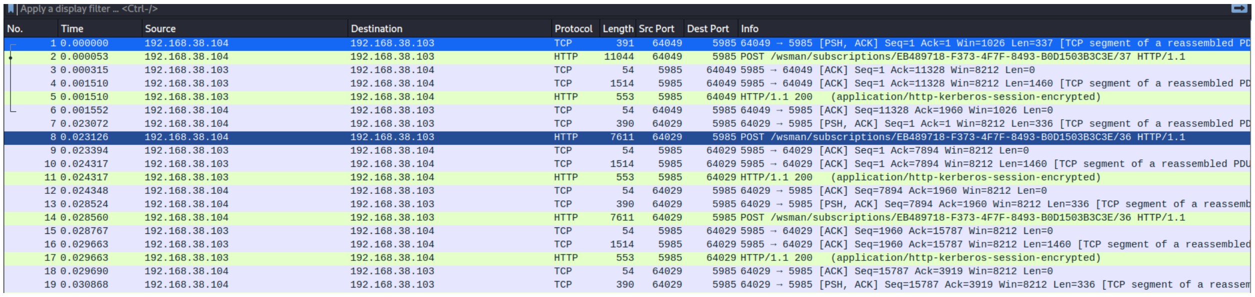
CTFTED2021{simple_profile_creation_and_fake_password}

Network Analyze

Baby Shark

Langkah Penyelesaian:

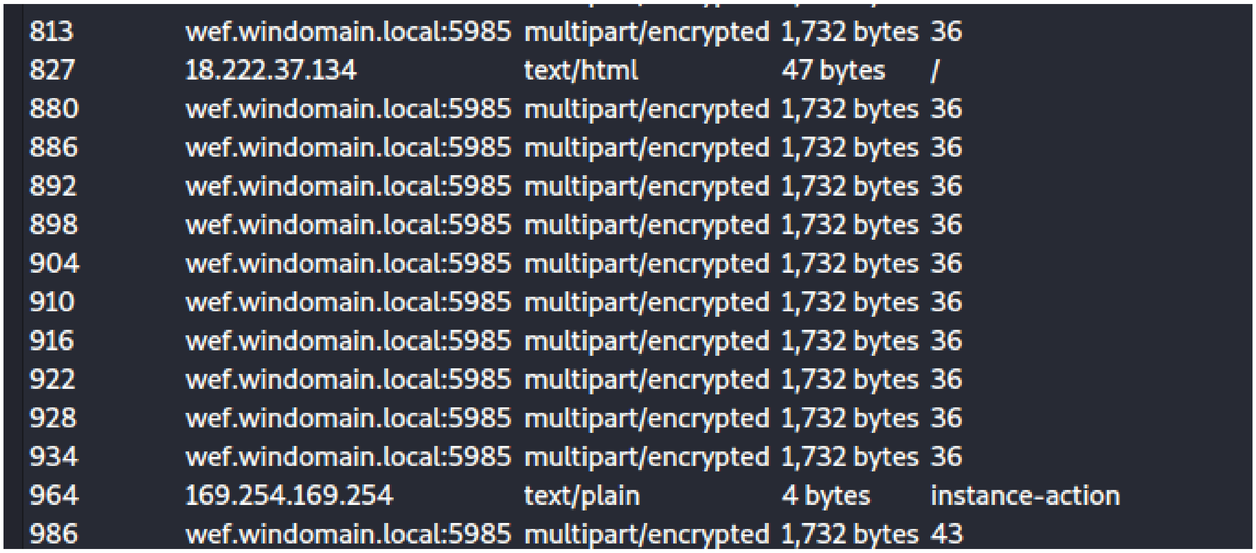
Diberikan sebuah PCAP yang awalnya terlihat mengerikan ada encrypted data



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Src Port	Dest Port	Info
1	0.000000	192.168.38.104	192.168.38.103	TCP	391	64049	5985	64049 → 5985 [PSH, ACK] Seq=1 Ack=1 Win=1026 Len=337 [TCP segment of a reassembled PD
2	0.000053	192.168.38.104	192.168.38.103	HTTP	11044	64049	5985	POST /wsman/subscriptions/EB489718-F373-4F7F-8493-B0D1503B3C3E/37 HTTP/1.1
3	0.000315	192.168.38.103	192.168.38.104	TCP	54	5985	64049	5985 → 64049 [ACK] Seq=1 Ack=11328 Win=8212 Len=0
4	0.001510	192.168.38.103	192.168.38.104	TCP	1514	5985	64049	5985 → 64049 [ACK] Seq=1 Ack=11328 Win=8212 Len=1460 [TCP segment of a reassembled PD
5	0.001510	192.168.38.103	192.168.38.104	HTTP	553	5985	64049	HTTP/1.1 200 (application/http-kerberos-session-encrypted)
6	0.001552	192.168.38.104	192.168.38.103	TCP	54	64049	5985	64049 → 5985 [ACK] Seq=11328 Ack=1960 Win=1026 Len=0
7	0.023072	192.168.38.104	192.168.38.103	TCP	390	64029	5985	64029 → 5985 [PSH, ACK] Seq=1 Ack=1 Win=8212 Len=336 [TCP segment of a reassembled PD
8	0.023126	192.168.38.104	192.168.38.103	HTTP	7611	64029	5985	POST /wsman/subscriptions/EB489718-F373-4F7F-8493-B0D1503B3C3E/36 HTTP/1.1
9	0.023394	192.168.38.103	192.168.38.104	TCP	54	5985	64029	5985 → 64029 [ACK] Seq=1 Ack=7894 Win=8212 Len=0
10	0.024317	192.168.38.103	192.168.38.104	TCP	1514	5985	64029	5985 → 64029 [ACK] Seq=1 Ack=7894 Win=8212 Len=1460 [TCP segment of a reassembled PDU
11	0.024317	192.168.38.103	192.168.38.104	HTTP	553	5985	64029	HTTP/1.1 200 (application/http-kerberos-session-encrypted)
12	0.024348	192.168.38.104	192.168.38.103	TCP	54	64029	5985	64029 → 5985 [ACK] Seq=7894 Ack=1960 Win=8212 Len=0
13	0.028524	192.168.38.104	192.168.38.103	TCP	390	64029	5985	64029 → 5985 [PSH, ACK] Seq=7894 Ack=1960 Win=8212 Len=336 [TCP segment of a reassemb
14	0.028560	192.168.38.104	192.168.38.103	HTTP	7611	64029	5985	POST /wsman/subscriptions/EB489718-F373-4F7F-8493-B0D1503B3C3E/36 HTTP/1.1
15	0.028767	192.168.38.103	192.168.38.104	TCP	54	5985	64029	5985 → 64029 [ACK] Seq=1960 Ack=15787 Win=8212 Len=0
16	0.029663	192.168.38.103	192.168.38.104	TCP	1514	5985	64029	5985 → 64029 [ACK] Seq=1960 Ack=15787 Win=8212 Len=1460 [TCP segment of a reassembled
17	0.029663	192.168.38.103	192.168.38.104	HTTP	553	5985	64029	HTTP/1.1 200 (application/http-kerberos-session-encrypted)
18	0.029690	192.168.38.104	192.168.38.103	TCP	54	64029	5985	64029 → 5985 [ACK] Seq=15787 Ack=3919 Win=8212 Len=0
19	0.030868	192.168.38.104	192.168.38.103	TCP	390	64029	5985	64029 → 5985 [PSH, ACK] Seq=15787 Ack=3919 Win=8212 Len=336 [TCP segment of a reassem

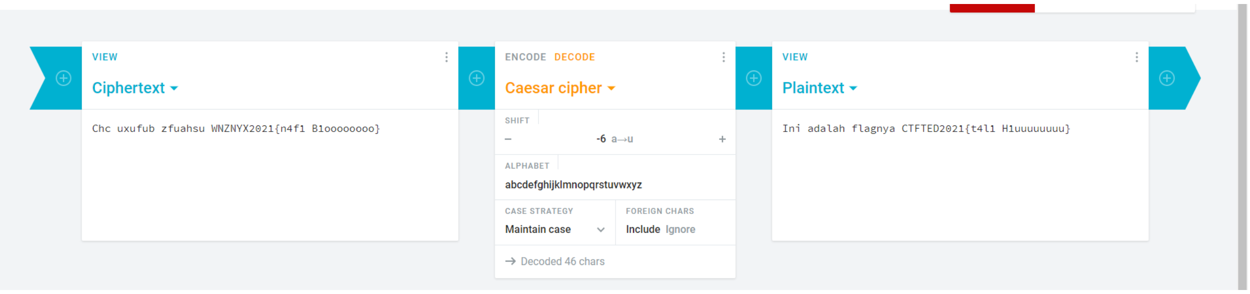
Ternyata dari export objects bisa terlihat satu page html



813	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
827	18.222.37.134	text/html	47 bytes	/
880	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
886	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
892	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
898	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
904	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
910	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
916	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
922	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
928	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
934	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	36
964	169.254.169.254	text/plain	4 bytes	instance-action
986	wef.windomain.local:5985	multipart/encrypted	1,732 bytes	43

Kalau di export dan di cat didapatkan Vav nqnynu syntaln PGSGRQ2021{g4y1 U1hhhhhhhh}

Tinggal di caesar cipher



```
Chc uxufub zfuahsu WNZNYX2021{n4f1 B1oooooooo}
```

SHIFT: -6 a→u

ALPHABET: abcdefghijklmnopqrstuvwxyz

CASE STRATEGY: Maintain case

FOREIGN CHARS: Include Ignore

→ Decoded 46 chars

```
Ini adalah flagnya CTFTED2021{t4l1 H1uuuuuuuu}
```

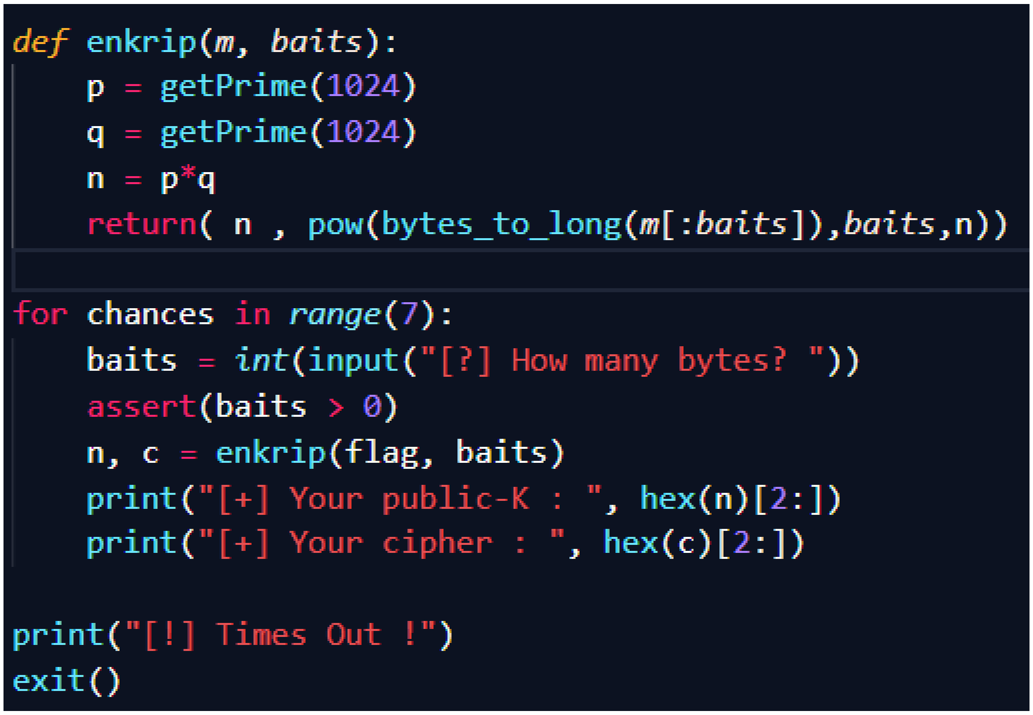
Flag:

```
CTFTED2021{t4l1 H1uuuuuuuu}
```

Cryptography

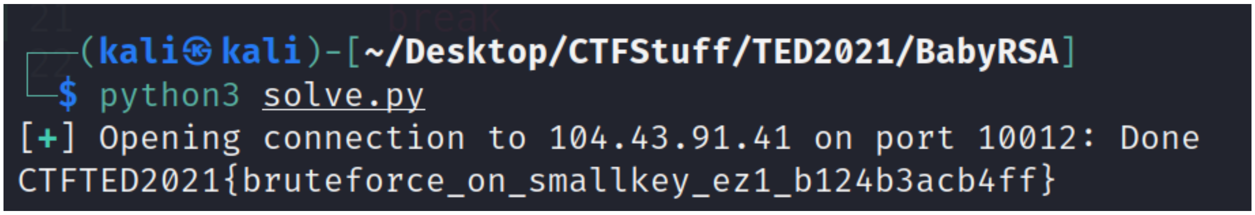
Baby RSA

Langkah Penyelesaian:



```python
def enkrip(m, baits):
    p = getPrime(1024)
    q = getPrime(1024)
    n = p*q
    return( n , pow(bytes_to_long(m[:baits]),baits,n))

for chances in range(7):
    baits = int(input("[?] How many bytes? "))
    assert(baits > 0)
    n, c = enkrip(flag, baits)
    print("[+] Your public-K : ", hex(n)[2:])
    print("[+] Your cipher : ", hex(c)[2:])

print("[!] Times Out !")
exit()
```

Diberikan soal seperti berikut, dari function encrypt nya kita bisa lihat bahwa kita bisa mengontrol exponent nya. Dari sana kita bisa coba satu per satu mengambil 1 bytes 1 bytes dan coba encrypt sendiri menggunakan public key yang diberikan.
Karena di limit percobaan nya 7 kali tinggal di ulang-ulang saja ditambahkan ke bagian flag hasil yang di temukan

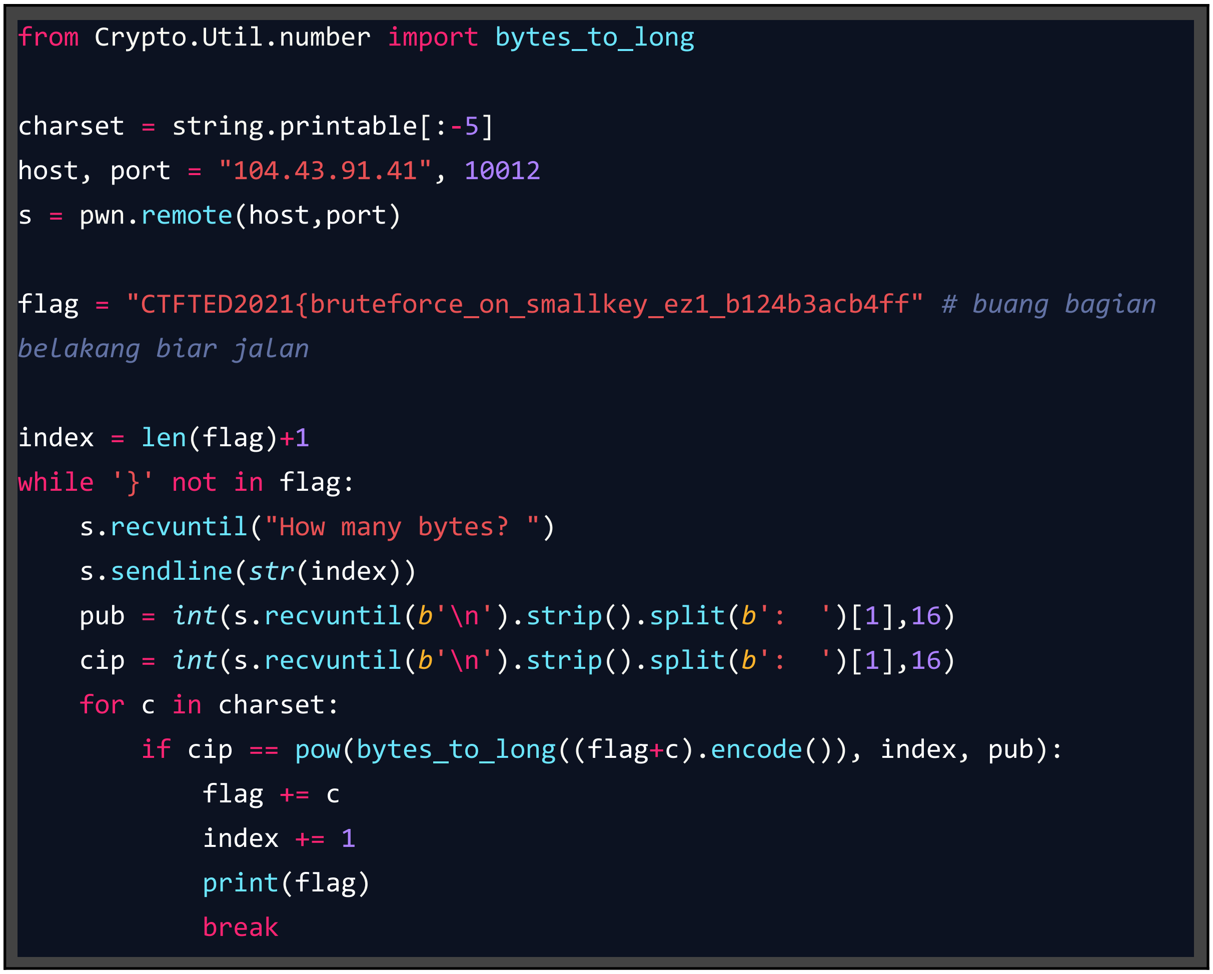


```
┌──(kali㉿kali)-[~/Desktop/CTFStuff/TED2021/BabyRSA]
└─$ python3 solve.py
[+] Opening connection to 104.43.91.41 on port 10012: Done
CTFTED2021{bruteforce_on_smallkey_ez1_b124b3acb4ff}
```

Code:

solve.py

```python
import pwn
import string
```

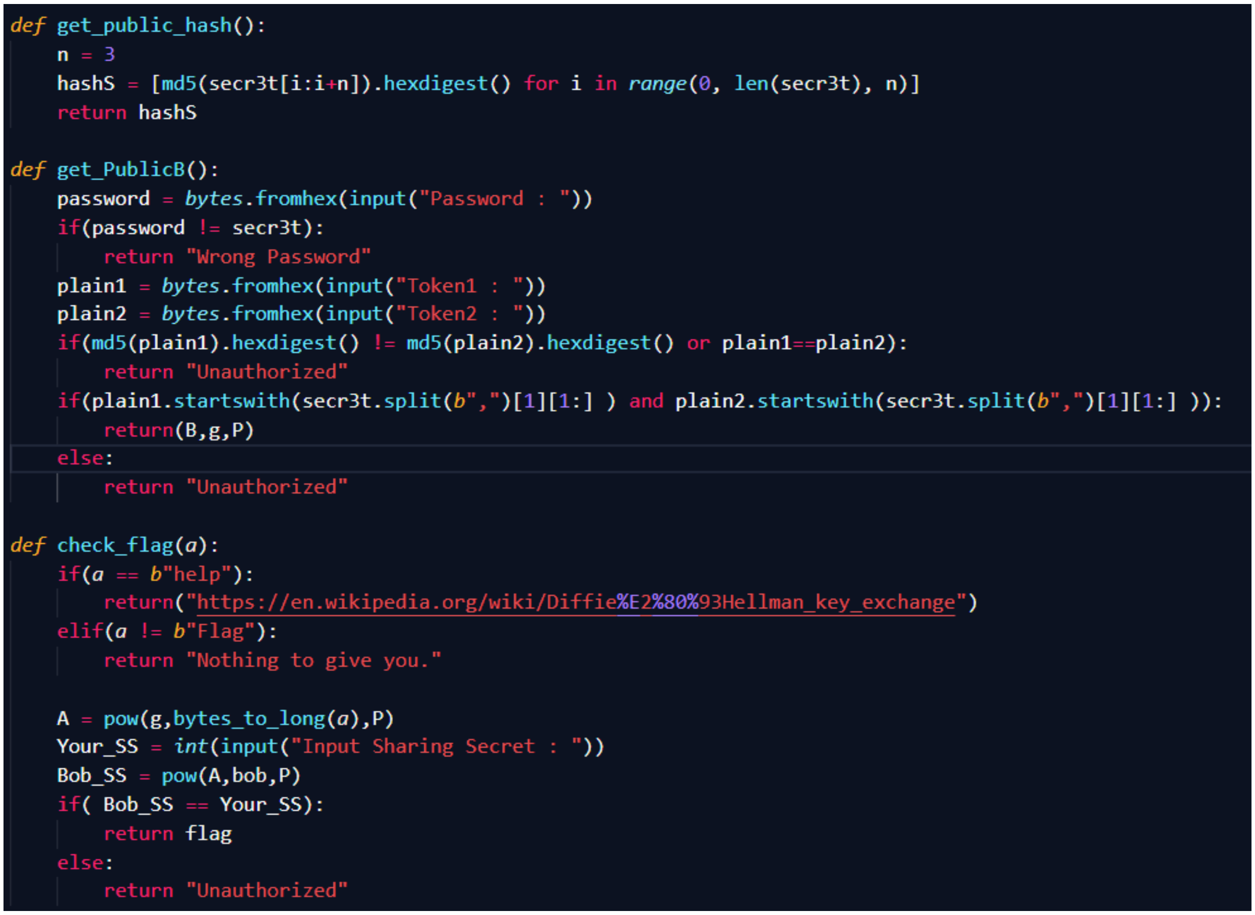


```python
from Crypto.Util.number import bytes_to_long

charset = string.printable[:-5]
host, port = "104.43.91.41", 10012
s = pwn.remote(host,port)

flag = "CTFTED2021{bruteforce_on_smallkey_ez1_b124b3acb4ff" # buang bagian belakang biar jalan

index = len(flag)+1
while '}' not in flag:
    s.recvuntil("How many bytes? ")
    s.sendline(str(index))
    pub = int(s.recvuntil(b'\n').strip().split(b':  ')[1],16)
    cip = int(s.recvuntil(b'\n').strip().split(b':  ')[1],16)
    for c in charset:
        if cip == pow(bytes_to_long((flag+c).encode()), index, pub):
            flag += c
            index += 1
            print(flag)
            break
```

Flag:
CTFTED2021{bruteforce_on_smallkey_ez1_b124b3acb4ff}

Baby Hash

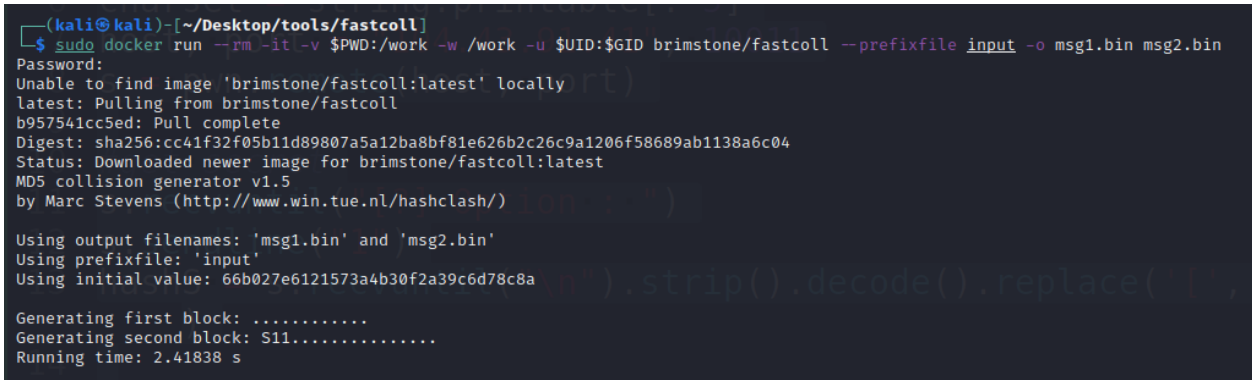
Langkah Penyelesaian:



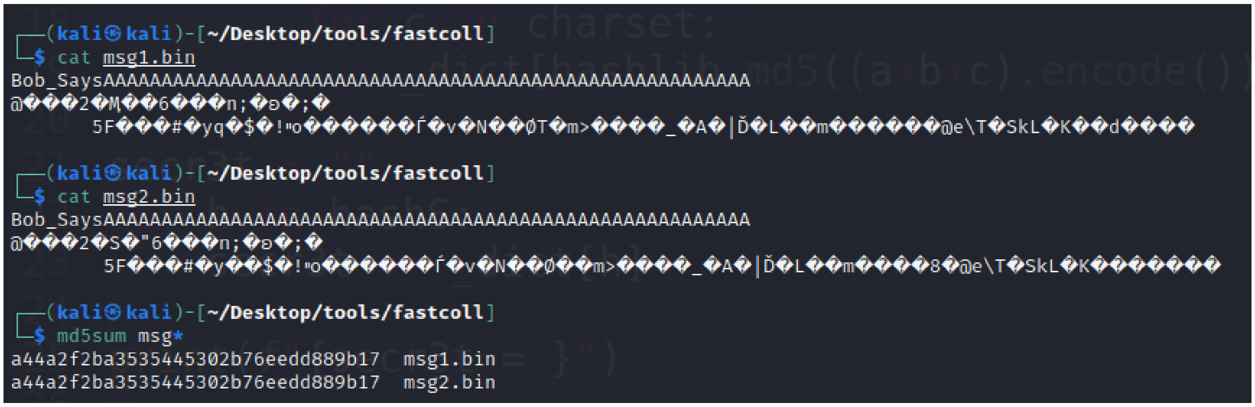
```
def get_public_hash():
    n = 3
    hashS = [md5(secr3t[i:i+n]).hexdigest() for i in range(0, len(secr3t), n)]
    return hashS

def get_PublicB():
    password = bytes.fromhex(input("Password : "))
    if(password != secr3t):
        return "Wrong Password"
    plain1 = bytes.fromhex(input("Token1 : "))
    plain2 = bytes.fromhex(input("Token2 : "))
    if(md5(plain1).hexdigest() != md5(plain2).hexdigest() or plain1==plain2):
        return "Unauthorized"
    if(plain1.startswith(secr3t.split(b",")[1][1:] ) and plain2.startswith(secr3t.split(b",")[1][1:] )):
        return(B,g,P)
    else:
        return "Unauthorized"

def check_flag(a):
    if(a == b"help"):
        return("https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange")
    elif(a != b"Flag"):
        return "Nothing to give you."

    A = pow(g,bytes_to_long(a),P)
    Your_SS = int(input("Input Sharing Secret : "))
    Bob_SS = pow(A,bob,P)
    if( Bob_SS == Your_SS):
        return flag
    else:
        return "Unauthorized"
```

Untuk soal yang kali ini kita ada 3 function yang perlu dilihat. get_public_hash, get_publicB, check_flag. Dari get_public_hash kita bisa mendapatkan secr3t yang ada dengan melakukan brute per 3 character yang di md5. Dari get_publicB kita bisa mendapatkan public component dan secret component "kita". Yang terakhir check_flag ini untuk mendapatkan flag. Mengikuti jalur itu kita pertama ambil secr3t, dan mencari hash collisions untuk get_publicB memakai prefix yang ada dari secret nya.

Mengikuti dokumentasi dari https://github.com/brimstone/fastcoll kita bisa mencari hash collisions nya dengan mudah.

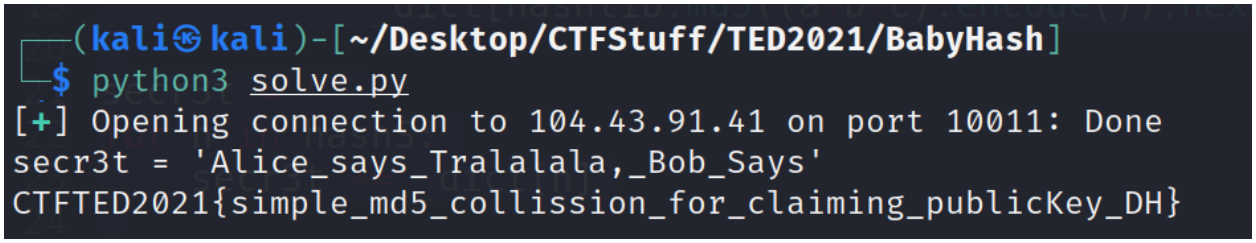


```
┌──(kali㉿kali)-[~/Desktop/tools/fastcoll]
└─$ sudo docker run --rm -it -v $PWD:/work -w /work -u $UID:$GID brimstone/fastcoll --prefixfile input -o msg1.bin msg2.bin
Password:
Unable to find image 'brimstone/fastcoll:latest' locally
latest: Pulling from brimstone/fastcoll
b957541cc5ed: Pull complete
Digest: sha256:cc41f32f05b11d89807a5a12ba8bf81e626b2c26c9a1206f58689ab1138a6c04
Status: Downloaded newer image for brimstone/fastcoll:latest
MD5 collision generator v1.5
by Marc Stevens (http://www.win.tue.nl/hashclash/)

Using output filenames: 'msg1.bin' and 'msg2.bin'
Using prefixfile: 'input'
Using initial value: 66b027e6121573a4b30f2a39c6d78c8a

Generating first block: ............
Generating second block: S11...............
Running time: 2.41838 s
```



```
┌──(kali㉿kali)-[~/Desktop/tools/fastcoll]
└─$ cat msg1.bin
Bob_SaysAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
@���2�M��6���n;�ə�;�
       5F���#�yq�$�!"o�������ŕ�v�N���ØT�m>����_�A�|Ď�L��m�������@e\T�SkL�K��d�����

┌──(kali㉿kali)-[~/Desktop/tools/fastcoll]
└─$ cat msg2.bin
Bob_SaysAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
@���2�S�"6���n;�ə�;�
        5F���#�y��$�!"o�������ŕ�v�N���Ø��m>����_�A�|Ď�L��m�����8�@e\T�SkL�K��������

┌──(kali㉿kali)-[~/Desktop/tools/fastcoll]
└─$ md5sum msg*
a44a2f2ba3535445302b76eedd889b17  msg1.bin
a44a2f2ba3535445302b76eedd889b17  msg2.bin
```

Setelah mengirimkan hash collisions ini, kita mendapatkan public component dan tinggal pakai itu untuk menyamakan sharingKey diffie hellman kita dengan bob.

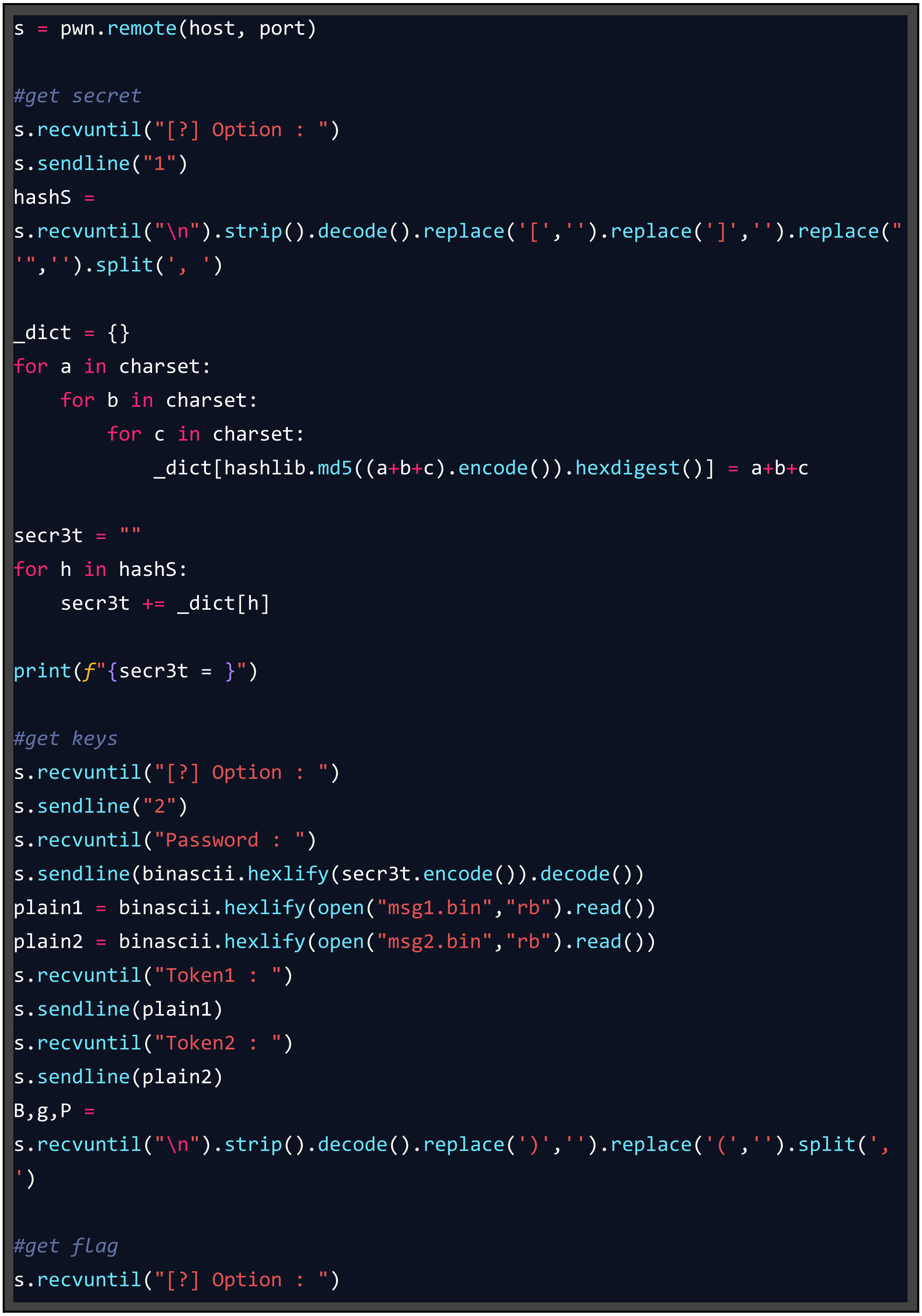


```
┌──(kali㉿kali)-[~/Desktop/CTFStuff/TED2021/BabyHash]
└─$ python3 solve.py
[+] Opening connection to 104.43.91.41 on port 10011: Done
secr3t = 'Alice_says_Tralalala,_Bob_Says'
CTFTED2021{simple_md5_collission_for_claiming_publicKey_DH}
```

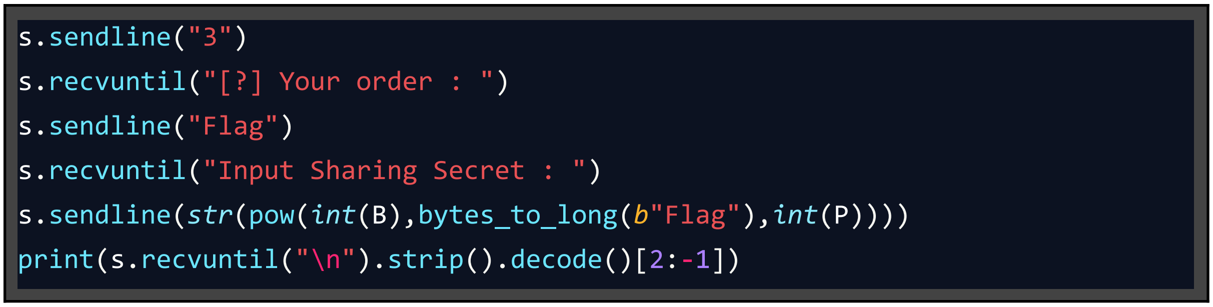
Code:

solve.py

```python
import hashlib
import string
import binascii
import pwn
from Crypto.Util.number import bytes_to_long
charset = string.printable[:-5]
host, port = "104.43.91.41", 10011
```



```python
s = pwn.remote(host, port)

#get secret
s.recvuntil("[?] Option : ")
s.sendline("1")
hashS = s.recvuntil("\n").strip().decode().replace('[','').replace(']','').replace("'",'').split(', ')

_dict = {}
for a in charset:
    for b in charset:
        for c in charset:
            _dict[hashlib.md5((a+b+c).encode()).hexdigest()] = a+b+c

secr3t = ""
for h in hashS:
    secr3t += _dict[h]

print(f"{secr3t = }")

#get keys
s.recvuntil("[?] Option : ")
s.sendline("2")
s.recvuntil("Password : ")
s.sendline(binascii.hexlify(secr3t.encode()).decode())
plain1 = binascii.hexlify(open("msg1.bin","rb").read())
plain2 = binascii.hexlify(open("msg2.bin","rb").read())
s.recvuntil("Token1 : ")
s.sendline(plain1)
s.recvuntil("Token2 : ")
s.sendline(plain2)
B,g,P = s.recvuntil("\n").strip().decode().replace(')','').replace('(','').split(', ')

#get flag
s.recvuntil("[?] Option : ")
```



```
s.sendline("3")
s.recvuntil("[?] Your order : ")
s.sendline("Flag")
s.recvuntil("Input Sharing Secret : ")
s.sendline(str(pow(int(B),bytes_to_long(b"Flag"),int(P))))
print(s.recvuntil("\n").strip().decode()[2:-1])
```

Flag:

CTFTED2021{simple_md5_collission_for_claiming_publicKey_DH}