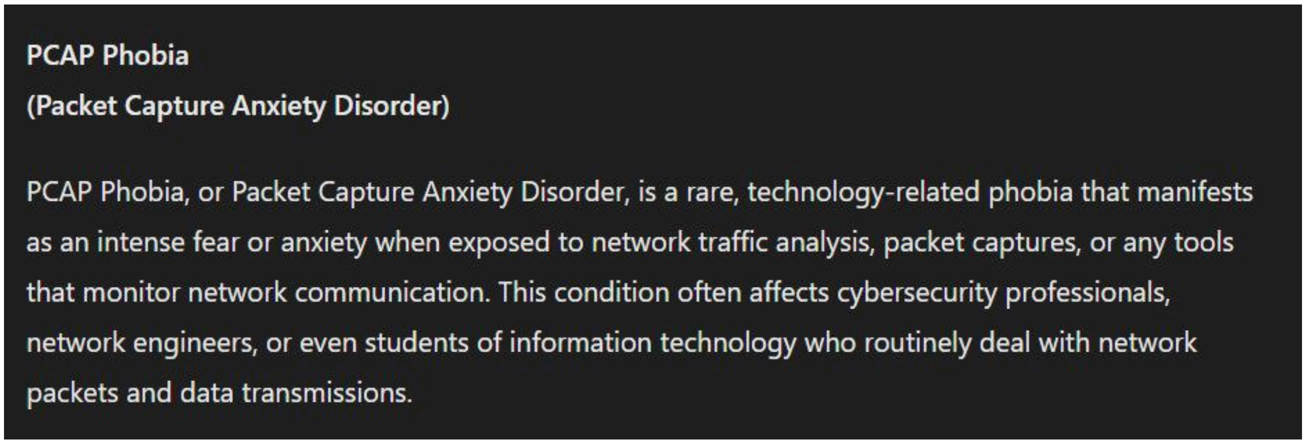
IFEST 2024 - Universitas Padjajaran

Played with SNI - FLAKEITO

k.eii's writeup
All Forensics + 1 Baby Web + 1 Rev + 1 Baby Cry

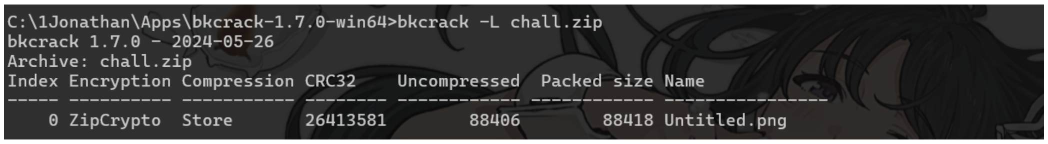


PCAP Phobia
(Packet Capture Anxiety Disorder)

PCAP Phobia, or Packet Capture Anxiety Disorder, is a rare, technology-related phobia that manifests as an intense fear or anxiety when exposed to network traffic analysis, packet captures, or any tools that monitor network communication. This condition often affects cybersecurity professionals, network engineers, or even students of information technology who routinely deal with network packets and data transmissions.

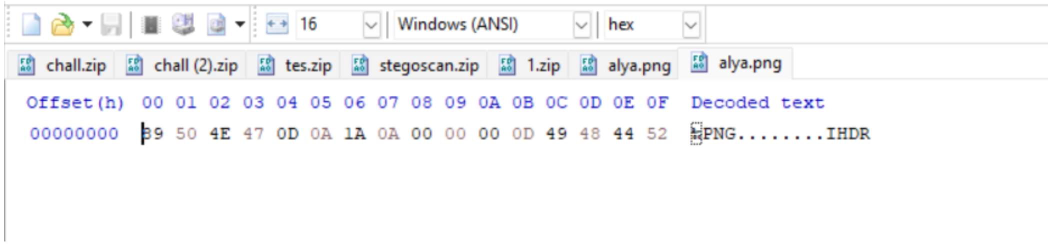
Forensic/Hari hari lupa password

(easy)
Coba crack pake john ga bisa, fcrack kelamaan, terus ane inget bkcrack.
(referensi: https://ctftime.org/writeup/15072)
Coba cek



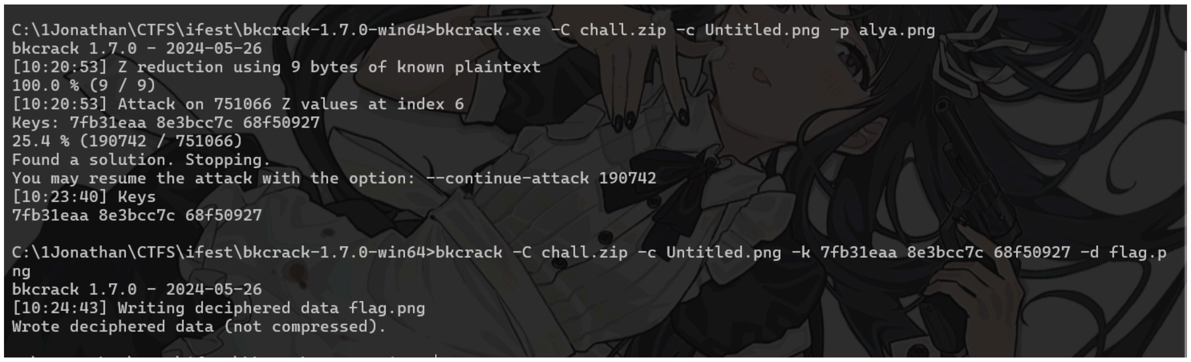
```
C:\1Jonathan\Apps\bkcrack-1.7.0-win64>bkcrack -L chall.zip
bkcrack 1.7.0 - 2024-05-26
Archive: chall.zip
Index Encryption Compression CRC32    Uncompressed  Packed size Name
----- ---------- ----------- -------- ------------ ------------ ----------------
    0 ZipCrypto  Store       26413581        88406        88418 Untitled.png
```

ZipCrypto rentan terhadap known plaintext attack, derived dari header chunk pngnya kita bisa attack
Disini di alya.png isinya header file doang, sebagai known plain textnya

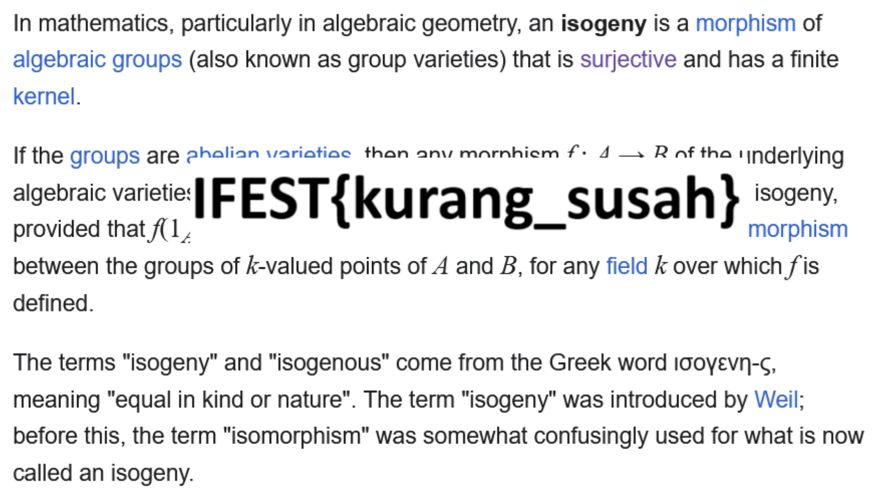


```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  ‰PNG........IHDR
```

(https://vincentandreas.medium.com/secretrezipe-zip-encryption-htb-writeup-51be4f816ce9)



```
C:\1Jonathan\CTFS\ifest\bkcrack-1.7.0-win64>bkcrack.exe -C chall.zip -c Untitled.png -p alya.png
bkcrack 1.7.0 - 2024-05-26
[10:20:53] Z reduction using 9 bytes of known plaintext
100.0 % (9 / 9)
[10:20:53] Attack on 751066 Z values at index 6
Keys: 7fb31eaa 8e3bcc7c 68f50927
25.4 % (190742 / 751066)
Found a solution. Stopping.
You may resume the attack with the option: --continue-attack 190742
[10:23:40] Keys
7fb31eaa 8e3bcc7c 68f50927

C:\1Jonathan\CTFS\ifest\bkcrack-1.7.0-win64>bkcrack -C chall.zip -c Untitled.png -k 7fb31eaa 8e3bcc7c 68f50927 -d flag.p
ng
bkcrack 1.7.0 - 2024-05-26
[10:24:43] Writing deciphered data flag.png
Wrote deciphered data (not compressed).
```



In mathematics, particularly in algebraic geometry, an **isogeny** is a morphism of algebraic groups (also known as group varieties) that is surjective and has a finite kernel.

If the groups are abelian varieties, then any morphism $f: A \to B$ of the underlying algebraic varietie[illegible] isogeny, provided that $f(1_A$ [illegible] morphism between the groups of k-valued points of A and B, for any field k over which f is defined.

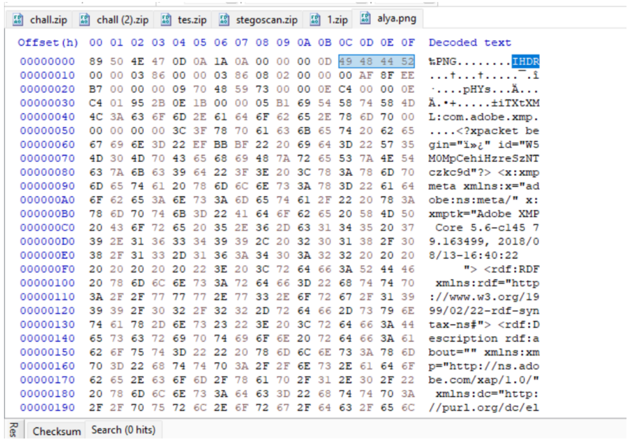
IFEST{kurang_susah}

The terms "isogeny" and "isogenous" come from the Greek word ισογενη-ς, meaning "equal in kind or nature". The term "isogeny" was introduced by Weil; before this, the term "isomorphism" was somewhat confusingly used for what is now called an isogeny.

Forensic/Tsundere Hex-chan

(baby)

Cuma hex fixing biasa

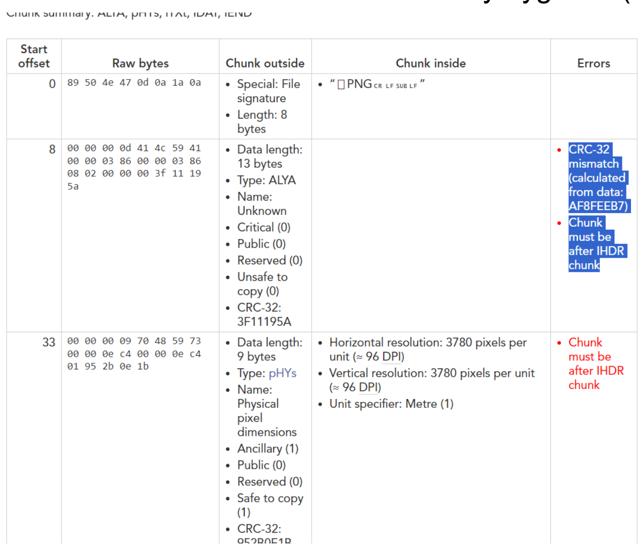


chall.zip | chall (2).zip | tes.zip | stegoscan.zip | 1.zip | alya.png

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text

00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  ‰PNG........IHDR
00000010  00 00 03 86 00 00 03 86 08 02 00 00 00 AF 8F EE  ...†...†.....¯.î
00000020  B7 00 00 00 09 70 48 59 73 00 00 0E C4 00 00 0E  ·....pHYs...Ä...
00000030  C4 01 95 2B 0E 1B 00 00 05 B1 69 54 58 74 58 4D  Ä.•+.....±iTXtXM
00000040  4C 3A 63 6F 6D 2E 61 64 6F 62 65 2E 78 6D 70 00  L:com.adobe.xmp.
00000050  00 00 00 00 3C 3F 78 70 61 63 6B 65 74 20 62 65  ....<?xpacket be
00000060  67 69 6E 3D 22 EF BB BF 22 20 69 64 3D 22 57 35  gin="ï»¿" id="W5
00000070  4D 30 4D 70 43 65 68 69 48 7A 72 65 53 7A 4E 54  M0MpCehiHzreSzNT
00000080  63 7A 6B 63 39 64 22 3F 3E 20 3C 78 3A 78 6D 70  czkc9d"?> <x:xmp
00000090  6D 65 74 61 20 78 6D 6C 6E 73 3A 78 3D 22 61 64  meta xmlns:x="ad
000000A0  6F 62 65 3A 6E 73 3A 6D 65 74 61 2F 22 20 78 3A  obe:ns:meta/" x:
000000B0  78 6D 70 74 6B 3D 22 41 64 6F 62 65 20 58 4D 50  xmptk="Adobe XMP
000000C0  20 43 6F 72 65 20 35 2E 36 2D 63 31 34 35 20 37   Core 5.6-c145 7
000000D0  39 2E 31 36 33 34 39 39 2C 20 32 30 31 38 2F 30  9.163499, 2018/0
000000E0  38 2F 31 33 2D 31 36 3A 34 30 3A 32 32 20 20 20  8/13-16:40:22
000000F0  20 20 20 20 20 22 3E 20 3C 72 64 66 3A 52 44 46       "> <rdf:RDF
00000100  20 78 6D 6C 6E 73 3A 72 64 66 3D 22 68 74 74 70   xmlns:rdf="http
00000110  3A 2F 2F 77 77 77 2E 77 33 2E 6F 72 67 2F 31 39  ://www.w3.org/19
00000120  39 39 2F 30 32 2F 32 32 2D 72 64 66 2D 73 79 6E  99/02/22-rdf-syn
00000130  74 61 78 2D 6E 73 23 22 3E 20 3C 72 64 66 3A 44  tax-ns#"> <rdf:D
00000140  65 73 63 72 69 70 74 69 6F 6E 20 72 64 66 3A 61  escription rdf:a
00000150  62 6F 75 74 3D 22 22 20 78 6D 6C 6E 73 3A 78 6D  bout="" xmlns:xm
00000160  70 3D 22 68 74 74 70 3A 2F 2F 6E 73 2E 61 64 6F  p="http://ns.ado
00000170  62 65 2E 63 6F 6D 2F 78 61 70 2F 31 2E 30 2F 22  be.com/xap/1.0/"
00000180  20 78 6D 6C 6E 73 3A 64 63 3D 22 68 74 74 70 3A   xmlns:dc="http:
00000190  2F 2F 70 75 72 6C 2E 6F 72 67 2F 64 63 2F 65 6C  //purl.org/dc/el
```

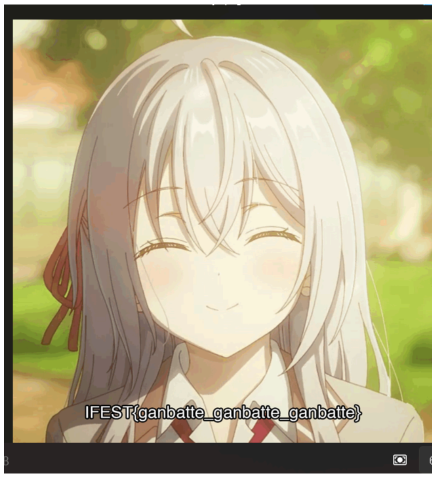
Checksum | Search (0 hits)

Fix IHDR chunk sama tadi ada CRCnya yg eror (bisa cek di nayuki.io)



Chunk summary: ALYA, pHYs, iTXt, IDAT, IEND

Start offset	Raw bytes	Chunk outside	Chunk inside	Errors
0	89 50 4e 47 0d 0a 1a 0a	• Special: File signature • Length: 8 bytes	• "PNG CR LF SUB LF"	
8	00 00 00 0d 41 4c 59 41 00 00 03 86 00 00 03 86 08 02 00 00 00 3f 11 19 5a	• Data length: 13 bytes • Type: ALYA • Name: Unknown • Critical (0) • Public (0) • Reserved (0) • Unsafe to copy (0) • CRC-32: 3F11195A		• CRC-32 mismatch (calculated from data: AF8FEEB7) • Chunk must be after IHDR chunk
33	00 00 00 09 70 48 59 73 00 00 0e c4 00 00 0e c4 01 95 2b 0e 1b	• Data length: 9 bytes • Type: pHYs • Name: Physical pixel dimensions • Ancillary (1) • Public (0) • Reserved (0) • Safe to copy (1) • CRC-32: 952B0E1B	• Horizontal resolution: 3780 pixels per unit (≈ 96 DPI) • Vertical resolution: 3780 pixels per unit (≈ 96 DPI) • Unit specifier: Metre (1)	• Chunk must be after IHDR chunk



Forensic/The Maestro

(medium)
Diberikan file .midi dan sc, yang kira2 adalah konsepnya kayak LSB
solv.py

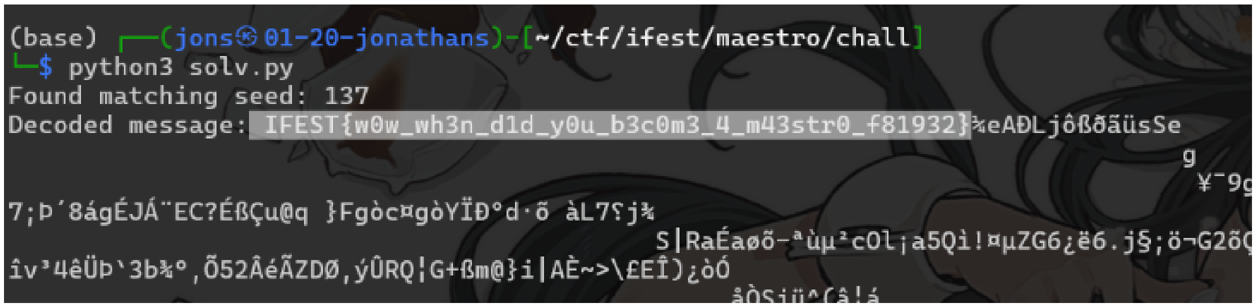
```
import mido
import random

def bits_to_text(bits):
    chars = []
    for i in range(0, len(bits), 8):
        byte = bits[i:i+8]
        chars.append(chr(int(byte, 2)))
    return ''.join(chars)

def decode_message_with_seed(midi_file, seed):
    mid = mido.MidiFile(midi_file)
    binary_message = []

    note_on_messages = []
    for track in mid.tracks:
        for msg in track:
            if msg.type == 'note_on':
                note_on_messages.append(msg)

    random.seed(seed)
    random.shuffle(note_on_messages)

    for msg in note_on_messages:
        lsb = msg.velocity & 1  # Extract the least significant bit of the
velocity
        binary_message.append(str(lsb))

    # Group the bits into bytes (8-bit groups) and convert to characters
    binary_message = ''.join(binary_message)

    if len(binary_message) % 8 != 0:
        binary_message = binary_message[:-(len(binary_message) % 8)]  #
Truncate extra bits if needed

    return bits_to_text(binary_message)
```

```python
def brute_force_decode(midi_file, max_seed=1337, target_prefix="IFEST"):
    for seed in range(max_seed + 1):
        decoded_message = decode_message_with_seed(midi_file, seed)
        if decoded_message.startswith(target_prefix):
            print(f"Found matching seed: {seed}")
            print(f"Decoded message: {decoded_message}")
            return decoded_message, seed
    print("No matching seed found within the range.")
    return None, None

# Usage
decoded_message, found_seed = brute_force_decode('maestro.mid',
max_seed=1337, target_prefix="IFEST")
if decoded_message:
    print(f"Message: {decoded_message}")
else:
    print("Flag not found.")
```

Pakai mido, dari sc, flag disisipin melalui velocity/kecepatan audionya, kita ekstrak dari situ pake fungsi yang mirip2 (dibawah ini sc aslinya).
Di solver kita bruteforce seednya karena ada seed yang dipake buat ngerand.

```python
    for msg in note_on_messages:
        if message_index < max_len:
            bit = int(binary_message[message_index])
            if bit == 1:
                msg.velocity = min(127, msg.velocity | 1)
            else:
                msg.velocity = msg.velocity & ~1
            message_index += 1
```

Result



```
(base) ┌──(jons㉿01-20-jonathans)-[~/ctf/ifest/maestro/chall]
└─$ python3 solv.py
Found matching seed: 137
Decoded message: IFEST{w0w_wh3n_d1d_y0u_b3c0m3_4_m43str0_f81932}¾eAÐLjôßðãüsSe
                                                                                  g
                                                                                   ¥¯9g
7;Þ´8ágÉJÁ¨EC?ÉßÇu@q }Fgòc¤gòYÏÐ°d·õ àL7ʕj¾
                                            S|RaÉaøõ-ªùµ²cOl¡a5Qì!¤µZG6¿ë6.j§;ö¬G2õÇ
îv³4êÜÞ`3b¾°,Õ52ÂéÃZDØ,ýÛRQ¦G+ßm@}i|AÈ~>\£EÎ)¿òÓ
                                                 åÒSiü^(â!á
```

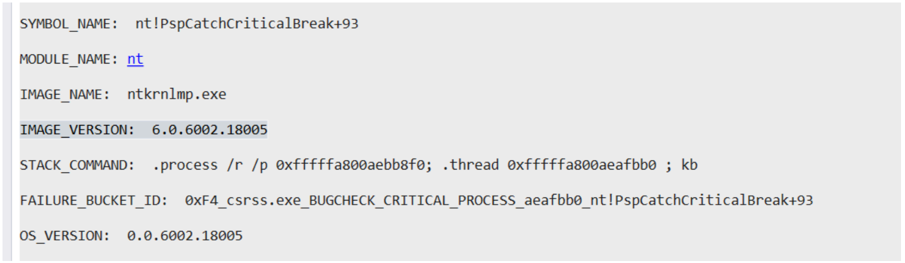
Forensic/Your PC ran into a problem and needs to restart

(hard)

Parser used: BlueScreenViewer & WinDbg (pakai WinDbg aja udah cukup sebenernya)
Ada semua sih di WinDbg

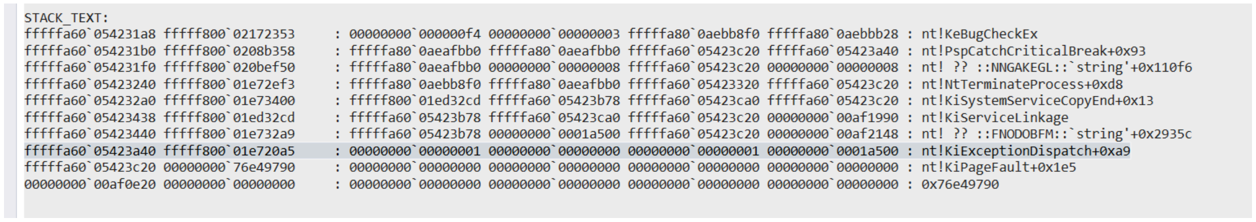
```
# 1. When was the dump captured? [mm/dd/yyyy]

# 2. What is the bug check code and its name? [0xcode:name]

# 3. What is the name of the terminated process?

# 4. What is the kernel version?

# 5. What is the address of the exceptions handler function? [0xaddress]

import struct
from pwn import *

p = remote('157.230.38.61', 11511)
p.recv()
ans = [
 '10/18/2009',
 '0xf4:CRITICAL_OBJECT_TERMINATION',
 'csrss.exe',
 '6.0.6002.18005',
 '0xfffff80001e71ffc' #01e72b40
 '',
]
for i in ans:
 p.sendlineafter(b'> ', i.encode())
 print(p.recv())
```

Kita bisa cek properti dari dump bsodnya pakai command “!analyze -v” di WinDbg



```
SYMBOL_NAME:  nt!PspCatchCriticalBreak+93

MODULE_NAME: nt

IMAGE_NAME:  ntkrnlmp.exe

IMAGE_VERSION:  6.0.6002.18005

STACK_COMMAND:  .process /r /p 0xfffffa800aebb8f0; .thread 0xfffffa800aeafbb0 ; kb

FAILURE_BUCKET_ID:  0xF4_csrss.exe_BUGCHECK_CRITICAL_PROCESS_aeafbb0_nt!PspCatchCriticalBreak+93

OS_VERSION:  0.0.6002.18005
```

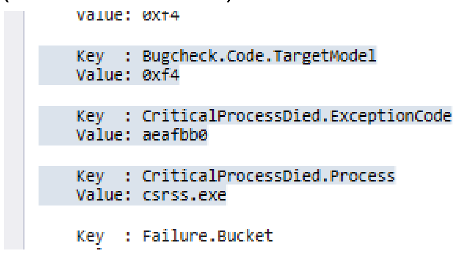
(kernel version)



```
STACK_TEXT:
fffffa60`054231a8 fffff800`02172353     : 00000000`000000f4 00000000`00000003 fffffa80`0aebb8f0 fffffa80`0aebbb28 : nt!KeBugCheckEx
fffffa60`054231b0 fffff800`0208b358     : fffffa80`0aeafbb0 fffffa80`0aeafbb0 fffffa60`05423c20 fffffa60`05423a40 : nt!PspCatchCriticalBreak+0x93
fffffa60`054231f0 fffff800`020bef50     : fffffa80`0aeafbb0 00000000`00000008 fffffa60`05423c20 00000000`00000008 : nt! ?? ::NNGAKEGL::`string'+0x110f6
fffffa60`05423240 fffff800`01e72ef3     : fffffa80`0aebb8f0 fffffa80`0aeafbb0 fffffa60`05423320 fffffa60`05423c20 : nt!NtTerminateProcess+0xd8
fffffa60`054232a0 fffff800`01e73400     : fffff800`01ed32cd fffffa60`05423b78 fffffa60`05423ca0 fffffa60`05423c20 : nt!KiSystemServiceCopyEnd+0x13
fffffa60`05423438 fffff800`01ed32cd     : fffffa60`05423b78 fffffa60`05423ca0 fffffa60`05423c20 00000000`00af1990 : nt!KiServiceLinkage
fffffa60`05423440 fffff800`01e732a9     : fffffa60`05423b78 00000000`0001a500 fffffa60`05423c20 00000000`00af2148 : nt! ?? ::FNODOBFM::`string'+0x2935c
fffffa60`05423a40 fffff800`01e720a5     : 00000000`00000001 00000000`00000000 00000000`00000001 00000000`0001a500 : nt!KiExceptionDispatch+0xa9
fffffa60`05423c20 00000000`76e49790     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : nt!KiPageFault+0x1e5
00000000`00af0e20 00000000`00000000     : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : 0x76e49790
```

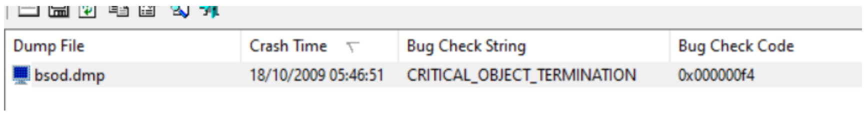
Retaddr - 0xa9 = `0xfffff80001e71ffc`

(Handler address)



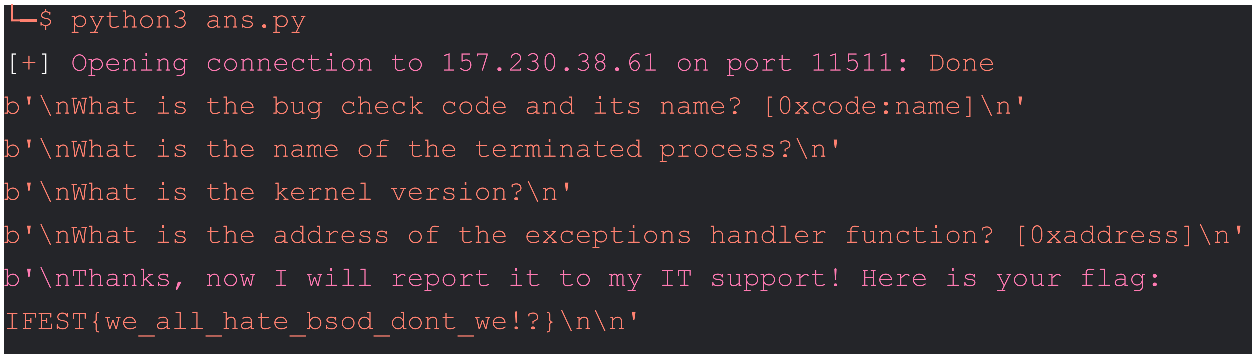
```
    Value: 0xf4

    Key  : Bugcheck.Code.TargetModel
    Value: 0xf4

    Key  : CriticalProcessDied.ExceptionCode
    Value: aeafbb0

    Key  : CriticalProcessDied.Process
    Value: csrss.exe

    Key  : Failure.Bucket
```

(Terminated process)



Dump File	Crash Time	Bug Check String	Bug Check Code
bsod.dmp	18/10/2009 05:46:51	CRITICAL_OBJECT_TERMINATION	0x000000f4

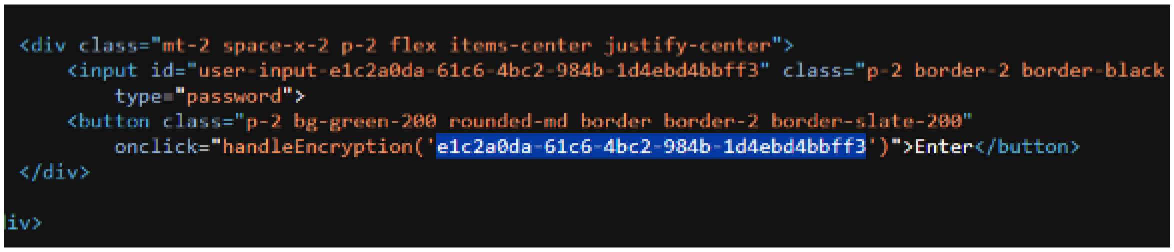
(yang ini pake BlueScreenViewer dapet bug check code, bug string, crash time)



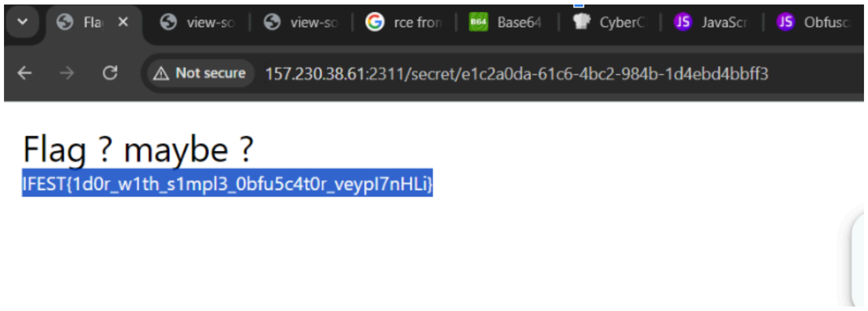
```
└─$ python3 ans.py
[+] Opening connection to 157.230.38.61 on port 11511: Done
b'\nWhat is the bug check code and its name? [0xcode:name]\n'
b'\nWhat is the name of the terminated process?\n'
b'\nWhat is the kernel version?\n'
b'\nWhat is the address of the exceptions handler function? [0xaddress]\n'
b'\nThanks, now I will report it to my IT support! Here is your flag:
IFEST{we_all_hate_bsod_dont_we!?}\n\n'
```

Web/Web Exploitation Sanity Check

(baby)



```
<div class="mt-2 space-x-2 p-2 flex items-center justify-center">
    <input id="user-input-e1c2a0da-61c6-4bc2-984b-1d4ebd4bbff3" class="p-2 border-2 border-black
        type="password">
    <button class="p-2 bg-green-200 rounded-md border border-2 border-slate-200"
        onclick="handleEncryption('e1c2a0da-61c6-4bc2-984b-1d4ebd4bbff3')">Enter</button>
</div>

iv>
```



```
157.230.38.61:2311/secret/e1c2a0da-61c6-4bc2-984b-1d4ebd4bbff3
```

Flag ? maybe ?
IFEST{1d0r_w1th_s1mpl3_0bfu5c4t0r_veypl7nHLi}

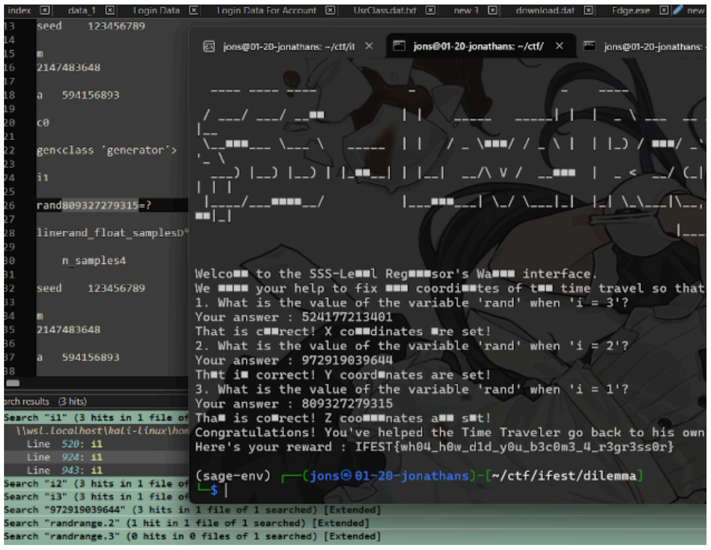
Rev/Time Traveler’s Dilemma

(easy)

Parsing dulu crunch tracenya dari byte ke char, disini yang nonprintable nggak tak keluarin biar rapih hasilnya

```python
#!/usr/bin/env python3
def is_printable(byte):
    return 32 <= byte <= 126 or byte in (9, 10, 13)

with open('session.chunked.pycrunch-trace', 'rb') as f:
    data = f.read()
printable_data = ''.join(chr(byte) for byte in data if is_printable(byte))

print(printable_data)
```

Sisanya aing cuma ctrl f i1 i2 i3 ntar ada value dari randnya



```
seed    123456789

m
2147483648

a   594156893

c0

gen<class 'generator'>

i1

rand809327279315=?

linerand_float_samplesD*

    n_samples4

seed    123456789

m
2147483648

a   594156893
```

```
Search "i1" (3 hits in 1 file of
  \\wsl.localhost\kali-linux\hom
    Line 520: i1
    Line 924: i1
    Line 943: i1
Search "i2" (3 hits in 1 file of
Search "i3" (3 hits in 1 file of
Search "972919039644" (3 hits in 1 file of 1 searched) [Extended]
Search "randrange.2" (1 hit in 1 file of 1 searched) [Extended]
Search "randrange.3" (0 hits in 0 files of 1 searched) [Extended]
```

```
Welco■■ to the SSS-Le■■l Reg■■■sor's Wa■■■ interface.
We ■■■■ your help to fix ■■■ coordi■■tes of t■■ time travel so that
1. What is the value of the variable 'rand' when 'i = 3'?
Your answer : 524177213401
That is c■■rect! X co■■dinates ■re set!
2. What is the value of the variable 'rand' when 'i = 2'?
Your answer : 972919039644
Th■t i■ correct! Y coord■nates are set!
3. What is the value of the variable 'rand' when 'i = 1'?
Your answer : 809327279315
Tha■ is co■rect! Z coo■■■nates a■■ s■t!
Congratulations! You've helped the Time Traveler go back to his own
Here's your reward : IFEST{wh04_h0w_d1d_y0u_b3c0m3_4_r3gr3ss0r}

(sage-env) ┌──(jons㉿01-20-jonathans)-[~/ctf/ifest/dilemma]
└─$ |
```

Crpyto/Aeshowspeed

(baby)

Diberikan aeshowspeed.py

Isinya skema yang mengenkripis flag.png menjadi flag.png.enc dengan AES-256 dalam mode CBC. terdapat padding pula dalam proses enkripsi sedangan IVnya melalui proses XOR dengan **0x10**

aeshowspeed.py

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
from cryptography.hazmat.backends import default_backend

def encrypt(file_path, key, iv):
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
    encryptor = cipher.encryptor()
    with open(file_path, "rb") as file:
        original_data = file.read()
    padding_length = 16 - len(original_data) % 16
    padded_data = original_data + bytes([padding_length] * padding_length)
    encrypted_data = encryptor.update(padded_data) + encryptor.finalize()
    encrypted_file_path = file_path + ".enc"
    with open(encrypted_file_path, "wb") as file:
        file.write(encrypted_data)
    return encrypted_file_path
key = b'IFEST2024mantapp'
key = key.ljust(32, b'\x35')
iv = key[:16]
iv = bytearray(iv)
for i in range(16):
    iv[i] = iv[i] ^ 0x10
iv = bytes(iv)
encrypt('flag.png',key,iv)
```

solver.py

```
from cryptography.hazmat.primitives.ciphers import Cipher, algorithms,
modes
from cryptography.hazmat.backends import default_backend

def decrypt(encrypted_file_path, key, iv):
    cipher = Cipher(algorithms.AES(key), modes.CBC(iv),
backend=default_backend())
    decryptor = cipher.decryptor()

    with open(encrypted_file_path, "rb") as file:
        encrypted_data = file.read()

    decrypted_data = decryptor.update(encrypted_data) +
decryptor.finalize()

    # Remove padding
    padding_length = decrypted_data[-1]
    original_data = decrypted_data[:-padding_length]

    original_file_path = encrypted_file_path.replace(".enc", "")
    with open(original_file_path, "wb") as file:
        file.write(original_data)

    return original_file_path

key = b'IFEST2024mantapp'
key = key.ljust(32, b'\x35')
iv = key[:16]
iv = bytearray(iv)
for i in range(16):
    iv[i] = iv[i] ^ 0x10
iv = bytes(iv)

decrypt('flag.png.enc', key, iv)
```