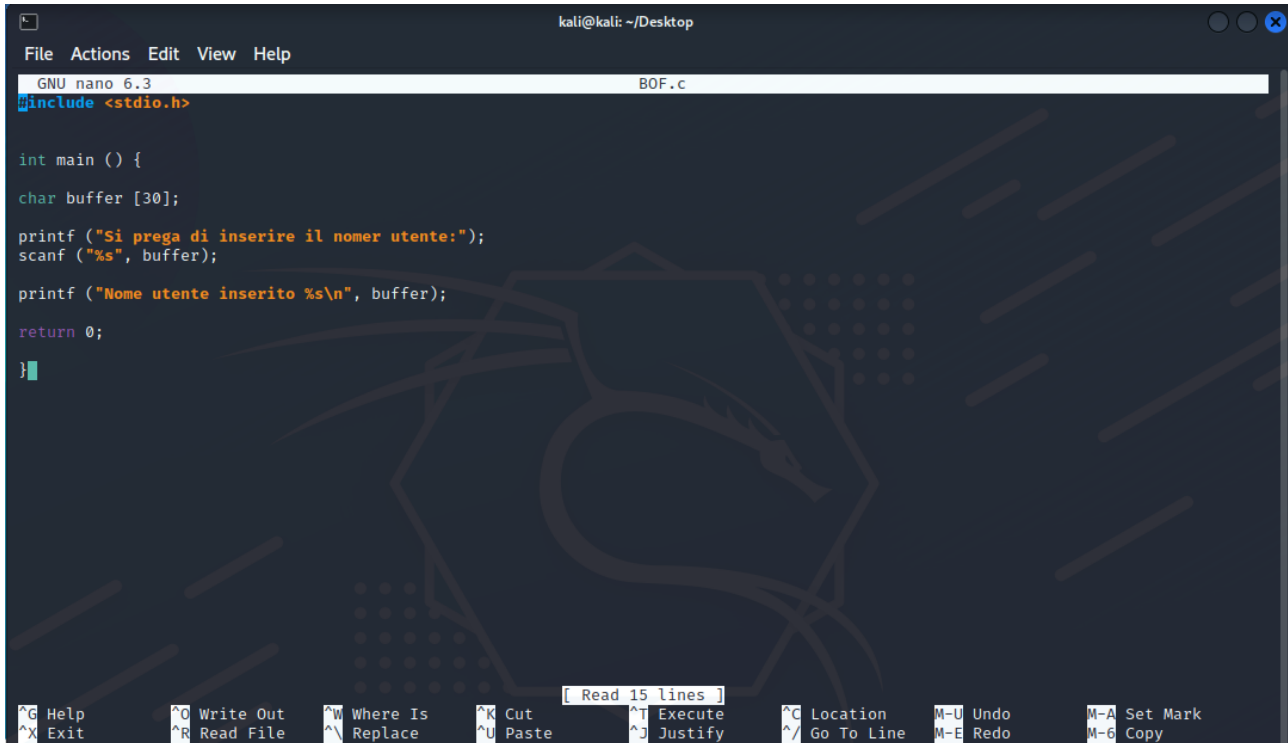


# BUFFER OVER FLOW

Nell'esercitazione di oggi abbiamo trattato del **Buffer Over Flow** e di come può interferire nel **Memory Managment**. Avendo modificato il codice, aumentando a [30] il buffer, ci siamo creati un alloggio di memoria superiore rispetto al codice iniziale.



```
kali@kali: ~/Desktop
GNU nano 6.3 BOF.c
#include <stdio.h>

int main () {
char buffer [30];

printf ("Si prega di inserire il numer utente:");
scanf ("%s", buffer);

printf ("Nome utente inserito %s\n", buffer);

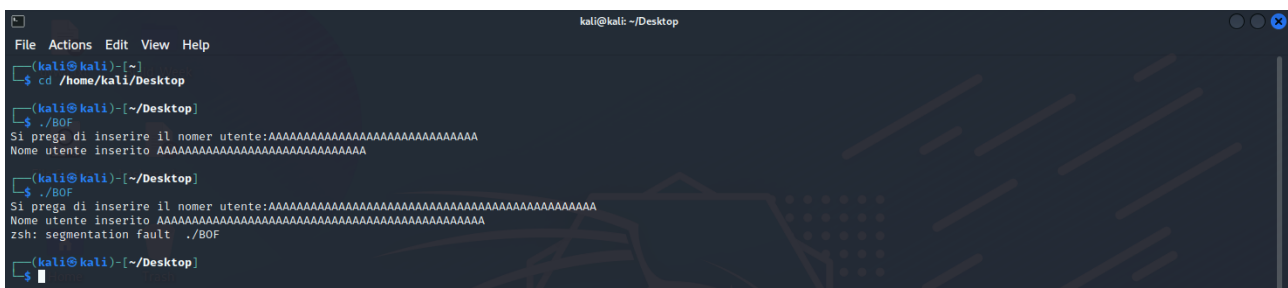
return 0;
}
```

## FASE 1

Nella prima fase andiamo a provare il nostro codice in linguaggio **C**. Fino a 30 caratteri, che equivalgono al valore del buffer, la nostra macchina stampa a video il valore del buffer, inserito con **%s**.

Se invece inseriamo troppi caratteri il programma si ferma per prevenire proprio un **Buffer over Flow**.

Ci stampa sempre a video i nostri 30 caratteri, perché quello “spazio” di memoria era stato “dedicato a noi” ma dopo crasha e non si chiude in maniera pulita. Questo comportamento avviene proprio per prevenire un **Buffer Over Flow**



```
kali@kali: ~/Desktop
File Actions Edit View Help
(kali@kali)~$ cd /home/kali/Desktop
(kali@kali)~/Desktop$ ./BOF
Si prega di inserire il numer utente:AAAAAAAAAAAAAAAAAAAAAAAAAAAA
Nome utente inserito AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
(kali@kali)~/Desktop$ ./BOF
Si prega di inserire il numer utente:AAAAAAAAAAAAAAAAAAAAAAAAAAAA
Nome utente inserito AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
zsh: segmentation fault ./BOF
(kali@kali)~/Desktop$
```