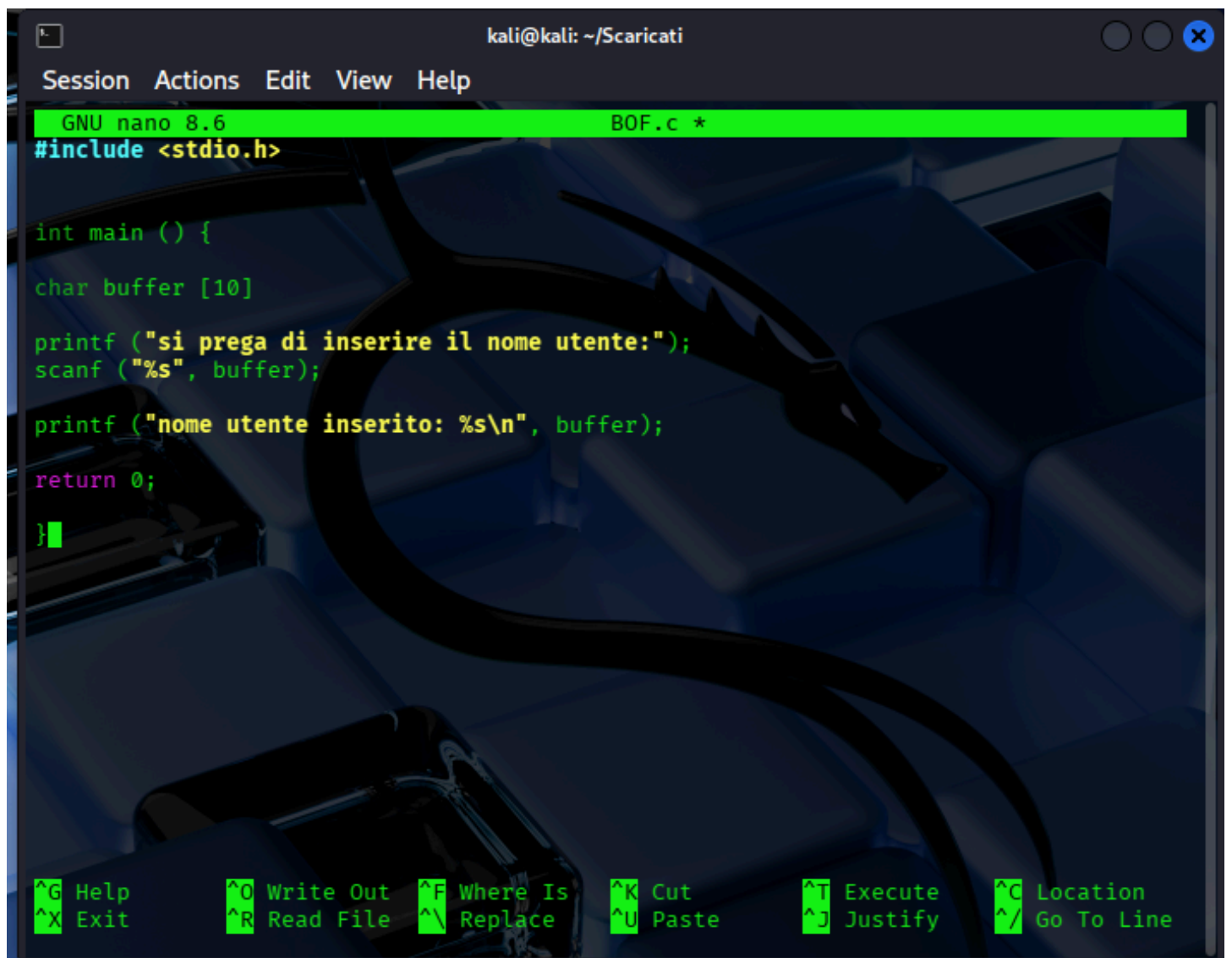


**Il buffer overflow è una vulnerabilità conseguenza di una mancanza di controllo dei limiti dei buffer che accettano input utente. Nelle prossime slide vedremo un esempio di codice in C volutamente vulnerabile ai BOF, e come scatenare una situazione di errore particolare chiamata «segmentation fault», ovvero un errore di memoria che si presenta quando un programma cerca inavvertitamente di scrivere su una posizione di memoria dove non gli è permesso scrivere (come può essere ad esempio una posizione di memoria dedicata a funzioni del sistema operativo).**

creiamo un documento con estensione in .c

utilizziamo il comando nano



```
kali@kali: ~/Scaricati
Session Actions Edit View Help
GNU nano 8.6 BOF.c *
#include <stdio.h>

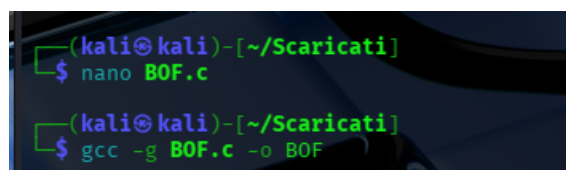
int main () {
char buffer [10]

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

printf ("nome utente inserito: %s\n", buffer);
return 0;
}

^G Help      ^O Write Out ^F Where Is  ^K Cut       ^T Execute
^X Exit      ^R Read File ^_ Replace   ^U Paste     ^J Justify
^C Location  ^/ Go To Line
```

compiliamo il file per poi lanciarlo



```
(kali@kali)-[~/Scaricati]
$ nano BOF.c

(kali@kali)-[~/Scaricati]
$ gcc -g BOF.c -o BOF
```

Inserendo un nome utente di 5 caratteri, il programma non ci riporta nessun problema, infatti come sappiamo il buffer accetta fino a 10 caratteri

```

(kali㉿kali)-[~/Scaricati]
$ ./BOF
si prega di inserire il nome utente:kali
nome utente inserito: kali

(kali㉿kali)-[~/Scaricati]
$ ./BOF
si prega di inserire il nome utente:fhuirbiuwrbguiwrbgiuwrbgirwbgrebguerbgierbgrebgiw
rbgwribgrowbgrowbgrbgb
nome utente inserito: fhuirbiuwrbguiwrbgiuwrbgirwbgrebguerbgierbgrebgiwrbgwribgrowbgr
owbgrbgb
zsh: segmentation fault ./BOF

```

Se inseriamo 30 caratteri il programma ci ritorna un errore, «segmentation fault», ovvero errore di segmentazione. L'errore di segmentazione avviene quando un programma, come abbiamo detto in precedenza, tenta di scrivere contenuti su una porzione di memoria alla quale non ha accesso. Questo è un chiaro esempio di BOF, abbiamo inserito 30 caratteri in un buffer che ne può contenere solamente 10 e di conseguenza alcuni caratteri stanno sovrascrivendo aree di memorie inaccessibili.

La traccia richiede di portare il vettore a 30 per verificare che sia possibile risolvere.

Modifichiamo il char buffet, compiliamo il file e vediamo che succede

```
char buffer [40];
```

```

(kali㉿kali)-[~/Scaricati]
$ nano BOF.c

(kali㉿kali)-[~/Scaricati]
$ gcc -g BOF.c -o BOF

(kali㉿kali)-[~/Scaricati]
$ ./BOF
si prega di inserire il nome utente:bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb
nome utente inserito: bbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb

(kali㉿kali)-[~/Scaricati]
$ █

```

soluzione trovata