

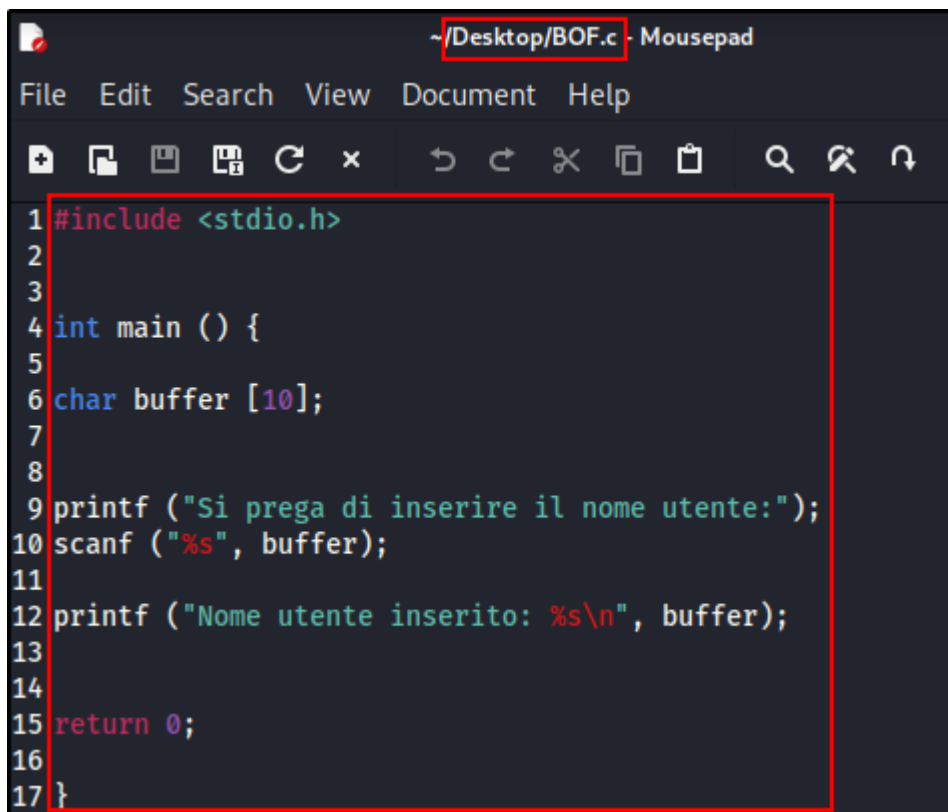
# Buffer overflow

## Descrizione esercizio

Nell'esercizio seguente, analizzeremo un frammento di codice in linguaggio C progettato intenzionalmente per essere vulnerabile a buffer overflow (BOF). Il nostro obiettivo sarà comprendere come l'esecuzione di questo codice può provocare un particolare errore di memoria noto come "segmentation fault". Questo errore si verifica quando il programma cerca inavvertitamente di scrivere in una posizione di memoria non autorizzata, come ad esempio quella dedicata alle funzioni del sistema operativo.

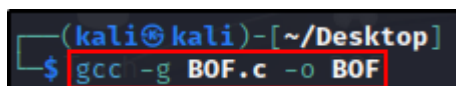
## Esecuzione

Ricopiamo il codice fornito e salviamolo sul Desktop in un file chiamato **BOF.c**

A screenshot of a text editor window titled "~//Desktop/BOF.c - Mousepad". The window has a menu bar with "File", "Edit", "Search", "View", "Document", and "Help". Below the menu is a toolbar with various icons. The main area contains C code with line numbers 1 through 17. The code is as follows:

```
1 #include <stdio.h>
2
3
4 int main () {
5
6 char buffer [10];
7
8
9 printf ("Si prega di inserire il nome utente:");
10 scanf ("%s", buffer);
11
12 printf ("Nome utente inserito: %s\n", buffer);
13
14
15 return 0;
16
17 }
```

Compiliamolo da terminale col seguente comando

A screenshot of a terminal window showing the command to compile the C program. The prompt is "(kali@kali)-[~/Desktop]" and the command entered is "gcc -g BOF.c -o BOF".

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

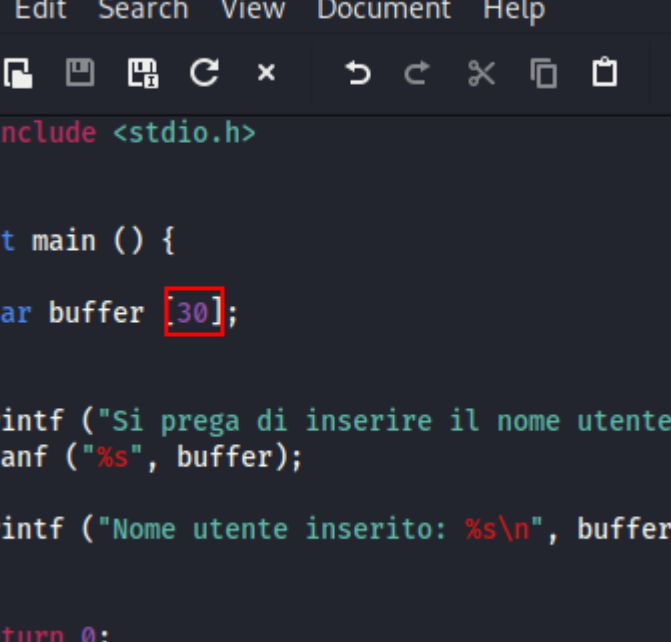
Avviamo il programma compilato e proviamo le due casistiche, prima inserendo un nome che rientra nel numero dei caratteri consentiti (10) e successivamente inserendo 30 caratteri

```
(kali㉿kali)-[~/Desktop]
$ ./BOF
Si prega di inserire il nome utente:Marco
Nome utente inserito: Marco
```

```
(kali㉿kali)-[~/Desktop]
$ ./BOF
Si prega di inserire il nome utente:123456789012345678901234567890
Nome utente inserito: 123456789012345678901234567890
zsh: segmentation fault ./BOF
```

Nel primo caso il programma esegue senza problemi, nel secondo riceviamo un errore di segmentazione, un esempio di buffer overflow

Adesso modifichiamo il codice cambiando la variabile che definisce il numero massimo di caratteri utilizzabili e salviamolo in un nuovo file chiamato **BOF30.c**



```
1 #include <stdio.h>
2
3
4 int main () {
5
6 char buffer [30];
7
8
9 printf ("Si prega di inserire il nome utente:");
10 scanf ("%s", buffer);
11
12 printf ("Nome utente inserito: %s\n", buffer);
13
14
15 return 0;
16
17 }
```

Dopo averlo compilato proviamo ad eseguire il file inserendo 30 caratteri

```
(kali㉿kali)-[~/Desktop]
$ ./BOF30
Si prega di inserire il nome utente:123456789012345678901234567890
Nome utente inserito: 123456789012345678901234567890
```

Adesso il programma si esegue senza errori

## Fix BOF

Un approccio per risolvere il problema del Buffer Overflow è quello di dare un numero massimo di caratteri che vengono letto dal buffer, questo si fa sostituendo `%s` dentro la funzione `scanf` mettendo il numero dei caratteri in mezzo ai due simboli come di seguito

```
1 #include <stdio.h>
2
3
4 int main () {
5
6 char buffer [10];
7
8
9 printf ("Si prega di inserire il nome utente:");
10 scanf ("%10s", buffer);
11
12 printf ("Nome utente inserito: %s\n", buffer);
13
14
15 return 0;
16
17 }
```

Andando ad eseguire il programma inserendo più di 10 caratteri vedremo che verranno visualizzati su schermo solo i primi 10

```
(kali㉿kali)-[~/Desktop]
$ ./BOFixed
Si prega di inserire il nome utente:1234567890123456789
Nome utente inserito: 1234567890
(kali㉿kali)-[~/Desktop]
$
```