

Obiettivo: Eseguire un Buffer Overflow su un file in C volutamente vulnerabile sulla macchina Kali e creare una situazione di errore particolare chiamata segmentatetion fault.

Prima Fase

Come prima cosa andiamo a creare il nostro codice in C utilizzando nano, posizionandoci sulla directory Desktop.

```
(kali㉿kali)-[~]  
$ cd /home/kali/Desktop  
  
(kali㉿kali)-[~/Desktop]  
$ sudo nano BOF.c  
[sudo] password for kali:
```

Inseriamo Char come variabile del buffer in quanto possiamo inserire qualsiasi carattere numerico o alfabetico, possiamo già notare che nelle parentesi quadre abbiamo la memoria dell'array buffer impostata a 10, cioè si potranno scrivere 10 caratteri.

```
File Actions Edit View Help  
GNU nano 6.4  
#include <stdio.h>  
  
int main(){  
char buffer [10];  
printf ("Inserire Nome utente: ");  
scanf ("%s", buffer);  
  
printf ("Nome utente: %s\n", buffer);  
  
return 0;  
}
```

Dopodiché andiamo a compilare il programma con il comando gcc lo switch -g e lo switch -o che in questo caso darà l'output sul nostro stesso codice.

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


Infine lanciamo il programma, scrivendo il mio nome composto da 8 caratteri il programma funziona correttamente.

```
(kali㉿kali)-[~/Desktop]  
$ ./BOF  
Inserire Nome utente: Gabriele  
Nome utente: Gabriele
```

Seconda Fase

Se invece andiamo a comporre più di 10 caratteri che come abbiamo visto sono stati impostati per la memoria di scrittura buffer del nostro codice, ci presenterà un errore (segmentation fault) cioè abbiamo superato lo spazio di memoria disponibile nel buffer e stiamo scrivendo nell'allocazione adiacente che non è destinata a questa operazione, quindi abbiamo causato un Buffer Overflow.

```
(kali㉿kali)-[~/Desktop]
$ ./BOF
Inserire Nome utente: GabrieleDigiampietro
Nome utente: GabrieleDigiampietro
zsh: segmentation fault ./BOF
```




Andiamo quindi a modificare il codice aumentando a 30 la memoria dell'array.

```
GNU nano 6.4
#include <stdio.h>

int main(){
char buffer [30];
printf ("Inserire Nome utente: ");
scanf ("%s", buffer);

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

return 0;
}
```



Infatti dopo avere modificato e ricompilato il nostro codice, inserendo il mio nome e cognome completo di 18 caratteri, il programma non dà errore perché stiamo usando la memoria messa a disposizione.

```
(kali㉿kali)-[~/Desktop]
$ ./BOF
Inserire Nome utente: GabrieleDigiampietro
Nome utente: GabrieleDigiampietro
```

Se invece proviamo a superare i 30 caratteri, ritroviamo l'errore e il conseguente Buffer Overflow.

```
(kali㉿kali)-[~/Desktop]
$ ./BOF
Inserire Nome utente: GabrieleDigiampietroaaaaaaaaabbbbbbbbbbbccccccccccccccbbbbbbb
Nome utente: GabrieleDigiampietroaaaaaaaaabbbbbbbbbbbccccccccccccccbbbbbbb
zsh: segmentation fault ./BOF
```

