

# BUFFER OVERFLOW

## Traccia:

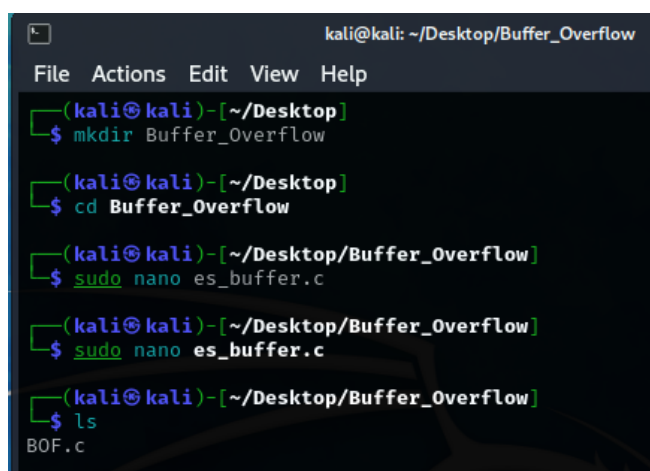
Nella lezione dedicata agli attacchi di sistema, abbiamo parlato dei buffer overflow, una vulnerabilità che è conseguenza di una mancanza di controllo dei limiti dei buffer che accettano input utente. Nelle prossime slide vedremo un esempio di codice in C volutamente vulnerabile ai BOF, e come scatenare 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).

Provate a riprodurre l'errore di segmentazione modificando il programma come di seguito:

- Aumentando la dimensione del vettore a 30.

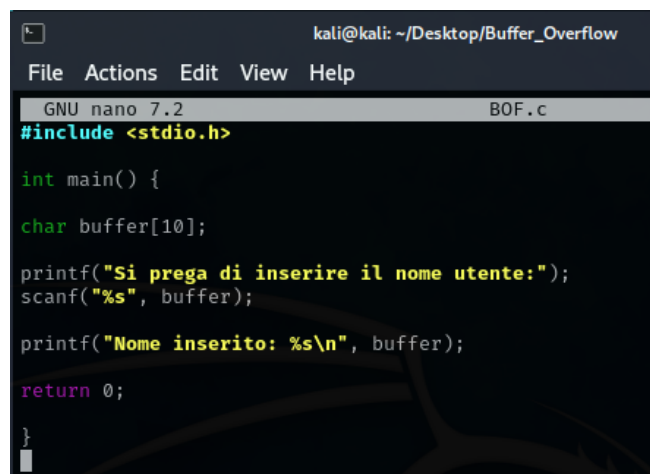
## Esecuzione:

Apriamo Kali-Linux e avviamo una shell. Per praticità ho creato una directory sul Desktop della macchina sfruttando la shell stessa. All'interno di questa, avviamo con il comando **sudo nano BOF.c** che crea da zero un file basato sul linguaggio C.



```
kali@kali: ~/Desktop/Buffer_Overflow
File Actions Edit View Help
(kali@kali)~[~/Desktop]
$ mkdir Buffer_Overflow
(kali@kali)~[~/Desktop]
$ cd Buffer_Overflow
(kali@kali)~[~/Desktop/Buffer_Overflow]
$ sudo nano es_buffer.c
(kali@kali)~[~/Desktop/Buffer_Overflow]
$ sudo nano es_buffer.c
(kali@kali)~[~/Desktop/Buffer_Overflow]
$ ls
BOF.c
```

All'interno di questo file, scriviamo il seguente codice.



```
kali@kali: ~/Desktop/Buffer_Overflow
File Actions Edit View Help
GNU nano 7.2 BOF.c
#include <stdio.h>

int main() {
    char buffer[10];

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

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

    return 0;
}
```

Scritto il sorgente, lo salviamo e lo compiliamo con il comando `gcc -g BOF.c -o BOF`. Così da creare il file eseguibile successivamente.

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

(kali㉿kali)-[~/Desktop/Buffer_Overflow]
$ ls
BOF  BOF.c
```

Eseguiamo il codice e proviamo ad inserire un nome utente nei limiti consentiti.

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

In questo caso non vi sono anomalie o segnalazione da parte del programma. Ma cosa succede se eccediamo il limite consentito di compilazione? Facciamo una prova.

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

Notiamo che il programma genera un errore di **“segmentation fault”**. Ciò si verifica quando si tenta di scrivere contenuti su una porzione di memoria alla quale non ha accesso. Questo è un chiaro esempio di BOF, abbiamo inserito più caratteri in un buffer che ne può contenere solamente 10 e di conseguenza alcuni caratteri stanno sovrascrivendo aree di memorie inaccessibili. Il valore 10 viene fuori dal codice che abbiamo creato in precedenza, che riportiamo di seguito.

```
char buffer[10];
```

Proviamo a modificare questo parametro con un valore più alto, tipo 30.

```
char buffer[30];
```

Lo salviamo, ricompiliamo il file e lo eseguiamo. Proviamo sia ad inserire una parola dentro i limiti programmati sia nel caso fosse in eccedenza.

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

(kali㉿kali)-[~/Desktop/Buffer_Overflow]
$ ./BOF
Si prega di inserire il nome utente:1234567890qwertyuiopasdfghjklzxcvbnm1234567890
Nome inserito: 1234567890qwertyuiopasdfghjklzxcvbnm1234567890
zsh: segmentation fault ./BOF
```

Come notiamo, nel primo tentativo questo passa senza problemi. Nel secondo questo genera un errore in quanto la parola inserita eccede il valore massimo consentito generando nuovamente l'errore. La differenza dal precedente sta che allungando il buffer di memoria la probabilità che avvenga un buffer overflow si abbassa. Se poi s'intende attaccare l'indice EIP, bisogna iniettare nel programma del codice che modifichi la posizione dell'indice durante l'esecuzione del programma stesso.