

Traccia:

Abbiamo già parlato del 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).

La vulnerabilità di tipo «**buffer overflow**» è generata da un mancato controllo, in fase di programmazione, dei limiti di un determinato buffer che accetta input utente. Un input maggiore della dimensione del buffer finisce per «sovrascrivere» il contenuto degli indirizzi di memoria adiacenti, modificando il contenuto potenzialmente delle variabili di un programma. Se correttamente sfruttata, una vulnerabilità di tipo Buffer Overflow (BOF), permette ad un utente malintenzionato di controllare il flusso del programma e potenzialmente eseguire codice malevolo.

Scriviamo un piccolo esempio di codice C volutamente vulnerabile.

```
File Edit Search View Document Help
1 #include <stdio.h>
2
3 int main () {
4     char buffer [10];
5
6     printf ("Inserire il nome utente, per favore: ");
7     scanf ("%s", buffer);
8
9     printf ("Hai inserito il seguente nome utente: %s\n", buffer);
10
11     return 0;
12 }
13 |
```

Compiliamo il file usando il comando `gcc -g BOF.c -o BOF`; ed eseguiamo il programma con il comando: `./BOF`

```
File Actions Edit View Help
(kali@kali)~[/Desktop]
$ gcc -g BOF.c -o BOF

(kali@kali)~[/Desktop]
$ ./BOF
Inserire il nome utente, per favore: Verdiana
Hai inserito il seguente nome utente: Verdiana

(kali@kali)~[/Desktop]
$ ./BOF
Inserire il nome utente, per favore: ahahahaahahahaahhahahahahahahahahah
Hai inserito il seguente nome utente: ahahahaahahahaahhahahahahahahahahah
zsh: segmentation fault ./BOF

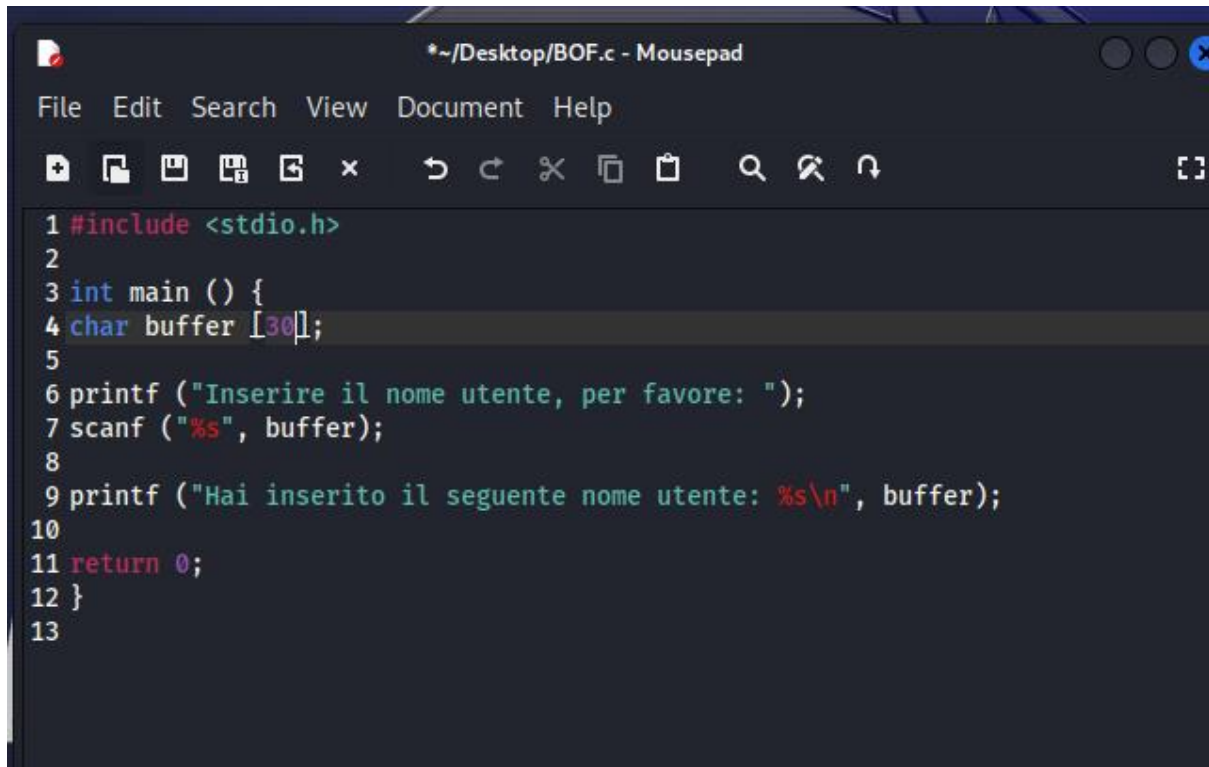
(kali@kali)~[/Desktop]
$
```

Il programma si avvia chiedendoci di inserire un nome utente: Inserendo un nome utente di 8 caratteri (Verdiana), il programma non ci riporta nessun problema.

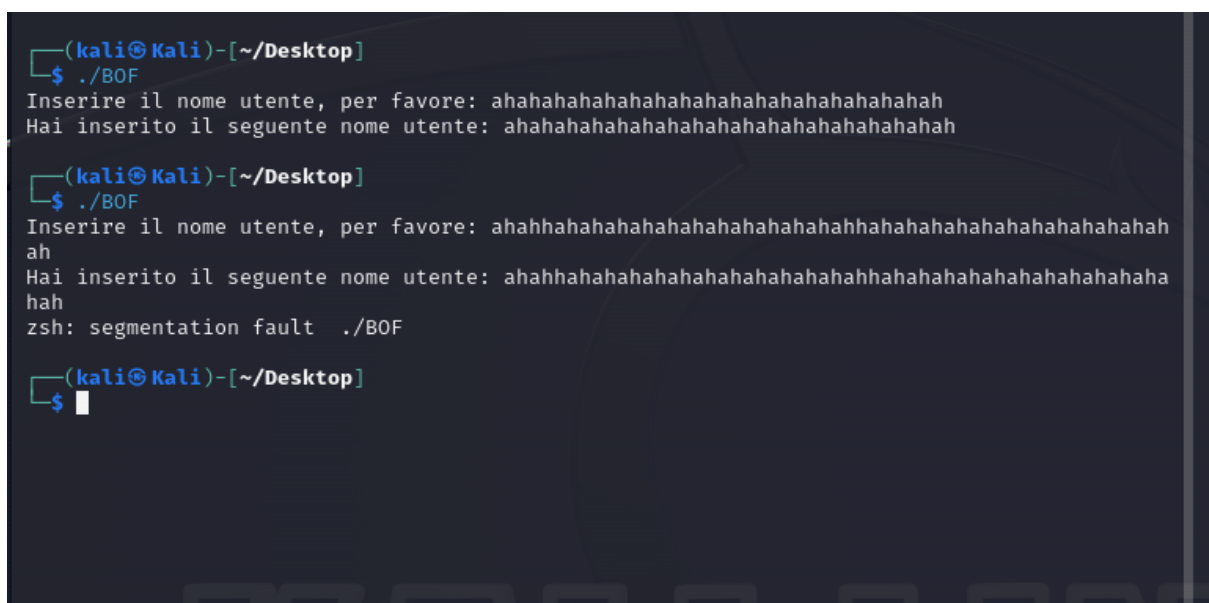
Cosa succede se inseriamo 30 caratteri?

Se inseriamo 30 caratteri il programma ci ritorna un errore, «**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.

Riproduciamo l'errore di segmentazione modificando aumentando la dimensione del vettore a 30.



```
*~/Desktop/BOF.c - Mousepad
File Edit Search View Document Help
+ [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons] [Icons]
1 #include <stdio.h>
2
3 int main () {
4 char buffer [30];
5
6 printf ("Inserire il nome utente, per favore: ");
7 scanf ("%s", buffer);
8
9 printf ("Hai inserito il seguente nome utente: %s\n", buffer);
10
11 return 0;
12 }
13
```



```
(kali㉿Kali)-[~/Desktop]
$ ./BOF
Inserire il nome utente, per favore: ahahahahahahahahahahahahahahahahah
Hai inserito il seguente nome utente: ahahahahahahahahahahahahahahahahah

(kali㉿Kali)-[~/Desktop]
$ ./BOF
Inserire il nome utente, per favore: ahahahahahahahahahahahahahahahahah
ah
Hai inserito il seguente nome utente: ahahahahahahahahahahahahahahahahah
hah
zsh: segmentation fault ./BOF

(kali㉿Kali)-[~/Desktop]
$
```