



ESERCIZIO W17D4

BUFFER OVERFLOW

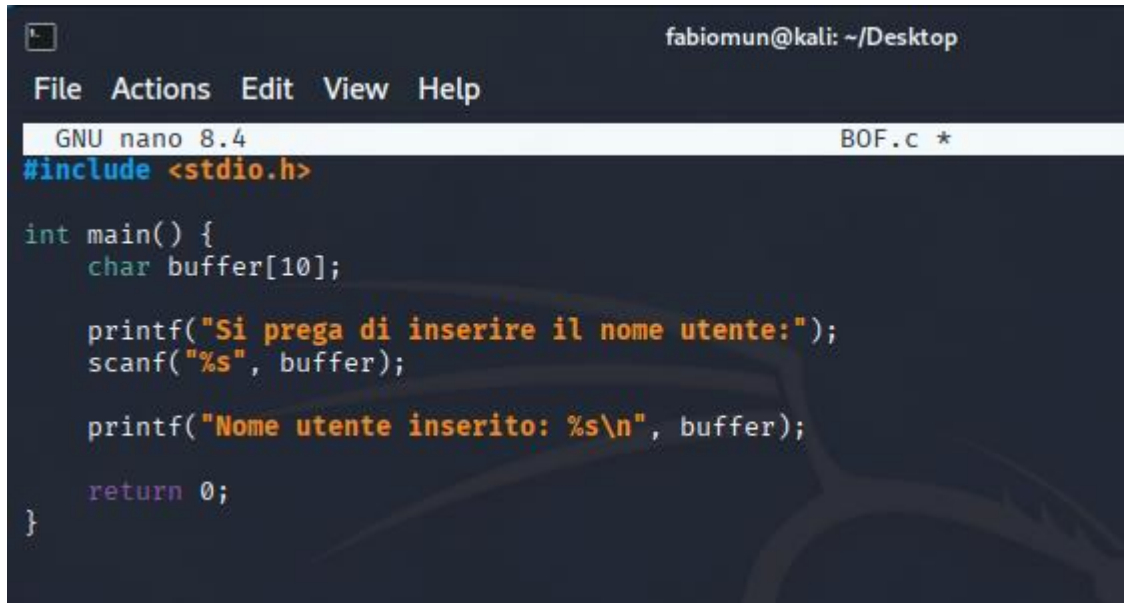
Mungiovì Fabio

TASK

In questa esercitazione andremo a scrivere un piccolo programma in C++, con l'obiettivo di capire il funzionamento del Buffer Overflow e le tecniche di mitigazione di questa vulnerabilità.

ESECUZIONE

Scriviamo questo semplice programma, dove viene richiesto l'inserimento di un nome utente all'interno di un array di 10 caratteri.



```
fabiomun@kali: ~/Desktop
File Actions Edit View Help
GNU nano 8.4 BOF.c *
#include <stdio.h>

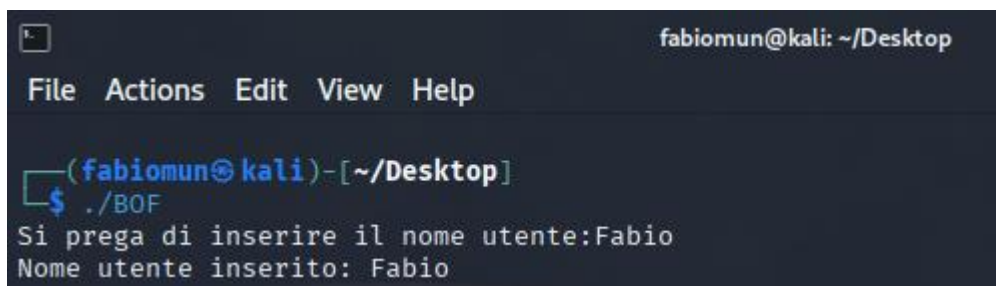
int main() {
    char buffer[10];

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

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

    return 0;
}
```

Una volta compilato lo avviamo e lo utilizziamo con un nome utente contenibile nell'array, come vediamo dall'immagine il programma restituisce l'output senza nessun problema.

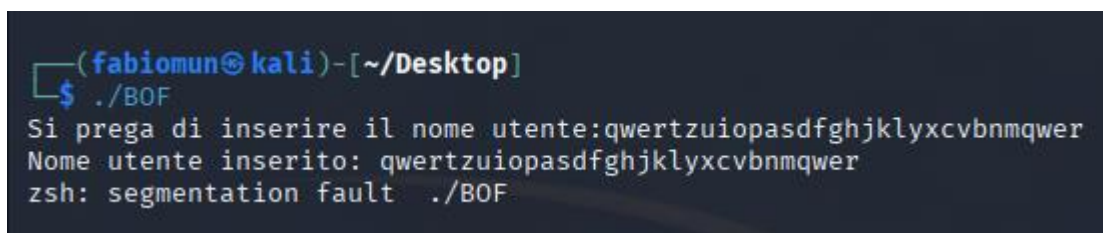


```
fabiomun@kali: ~/Desktop
File Actions Edit View Help

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

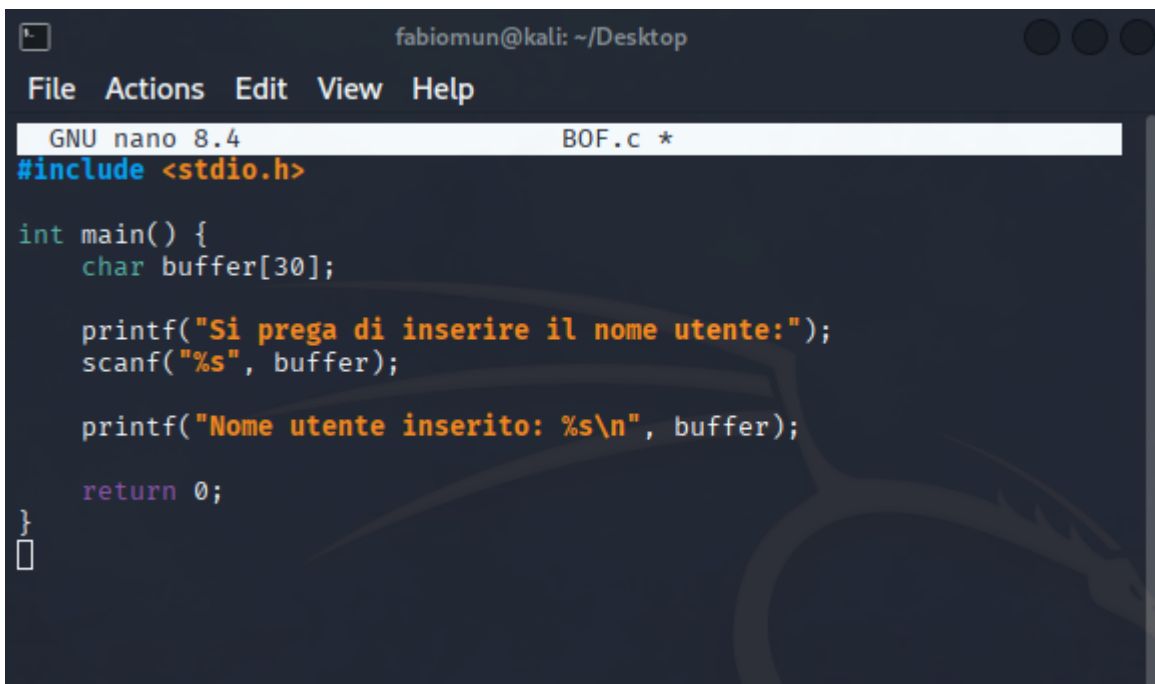
Proviamo ora a dare in input una stringa di 30 caratteri, come vediamo dall'immagine seguente il programma restituirà un errore, in quanto i caratteri inseriti sono troppi per l'array dichiarato in precedenza di 10 caratteri.

Questo è un esempio di **Buffer Overflow**



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

Proviamo adesso, modificando il codice del programma ad aumentare a 30 il numero di caratteri dell'array.



```
fabiomun@kali: ~/Desktop
File Actions Edit View Help
GNU nano 8.4 BOF.c *
#include <stdio.h>

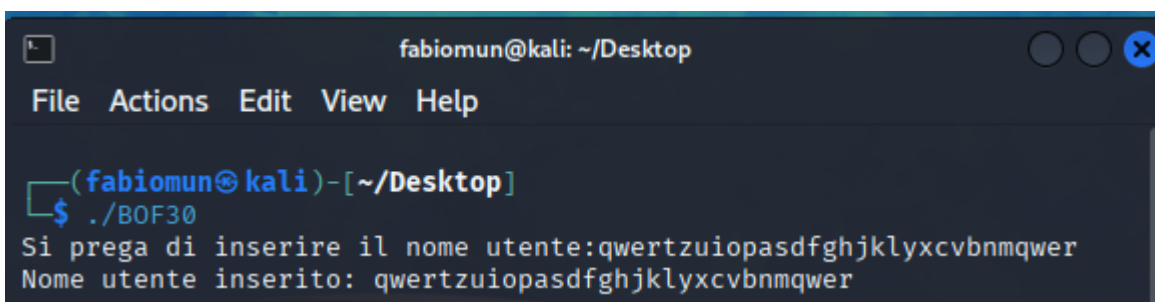
int main() {
    char buffer[30];

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

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

    return 0;
}
```

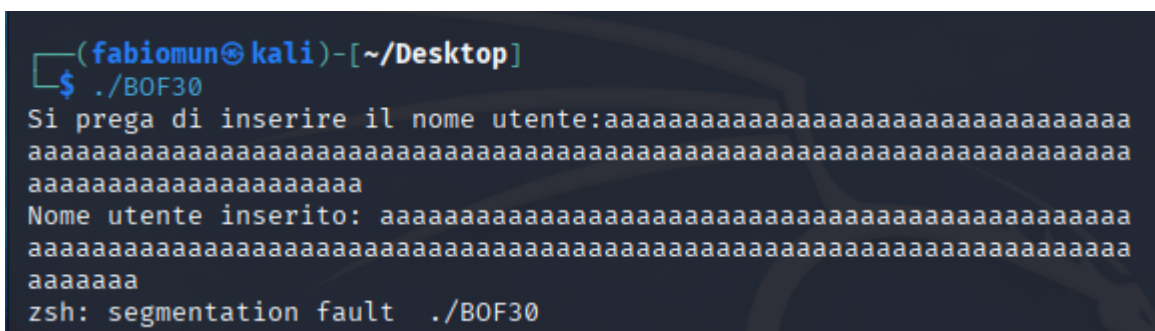
Inserendo ora una stringa di 30 caratteri vediamo che il programma non restituirà più l'errore precedente.



```
fabiomun@kali: ~/Desktop
File Actions Edit View Help

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

Ma il problema rimane comunque risolto parzialmente, in quanto se ora si inserisce una stringa con più di 30 caratteri, il problema del Buffer Overflow si ripresenta

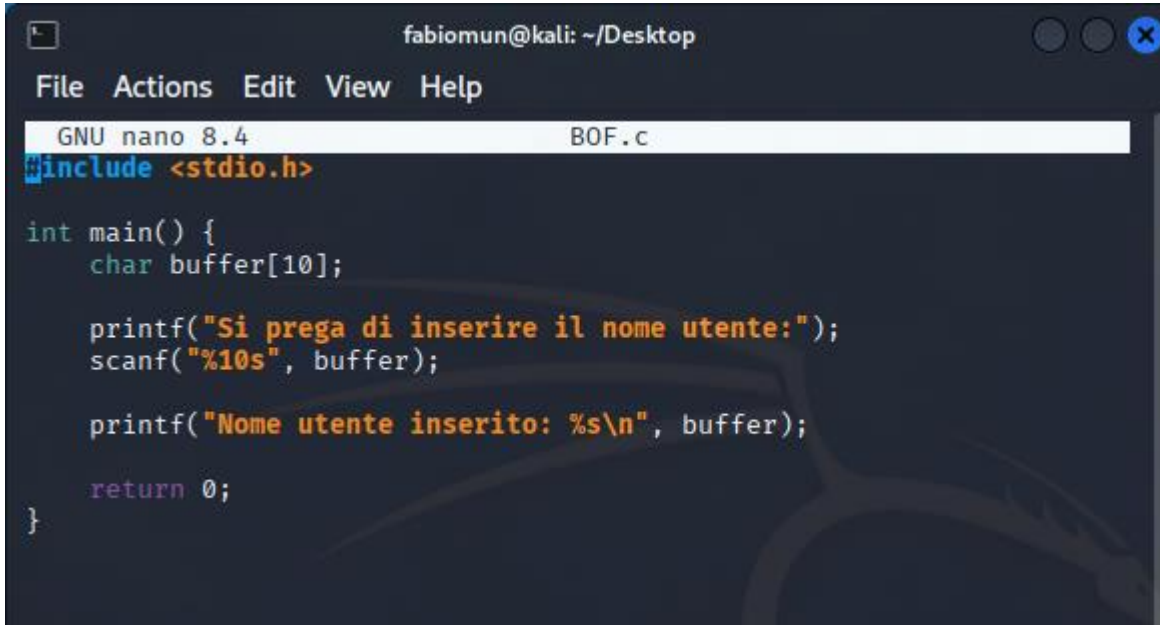


```
(fabiomun@kali)-[~/Desktop]
$ ./BOF30
Si prega di inserire il nome utente:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaa
Nome utente inserito: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaa
zsh: segmentation fault ./BOF30
```

MITIGAZIONE

Un metodo di mitigazione efficace nel nostro caso può essere l'inserimento di un controllo dell'input, che indica il numero massimo di caratteri che può leggere il programma.

Per fare questo aggiungiamo la parte di codice `%10s` all'interno dello scan dell'input, così che vengano presi in considerazione, e inseriti nell'array, solo i primi 10 caratteri inseriti dall'utente.



```
fabiomun@kali: ~/Desktop
File Actions Edit View Help
GNU nano 8.4 BOF.c
#include <stdio.h>

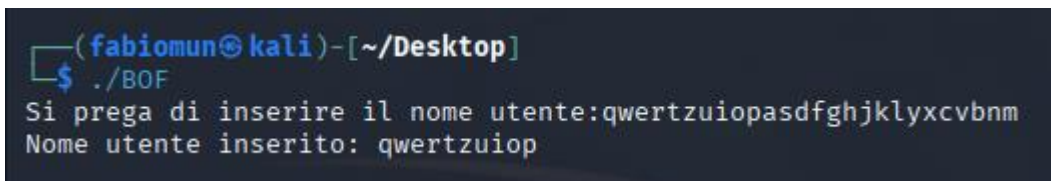
int main() {
    char buffer[10];

    printf("Si prega di inserire il nome utente:");
    scanf("%10s", buffer);

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

    return 0;
}
```

Avviamo quindi il programma e, come si vede nell'immagine seguente, inserendo una stringa molto lunga, il programma ci restituirà solo i primi 10 caratteri inseriti senza restituire nessun errore.



```
(fabiomun@kali)-[~/Desktop]
$ ./BOF
Si prega di inserire il nome utente:qwertyuiopasdfghjklxyxvbnm
Nome utente inserito: qwertyuiop
```

Questa è risultata essere, per il nostro caso, una efficiente tecnica di mitigazione del Buffer Overflow