

CONSEGNA S7/L4

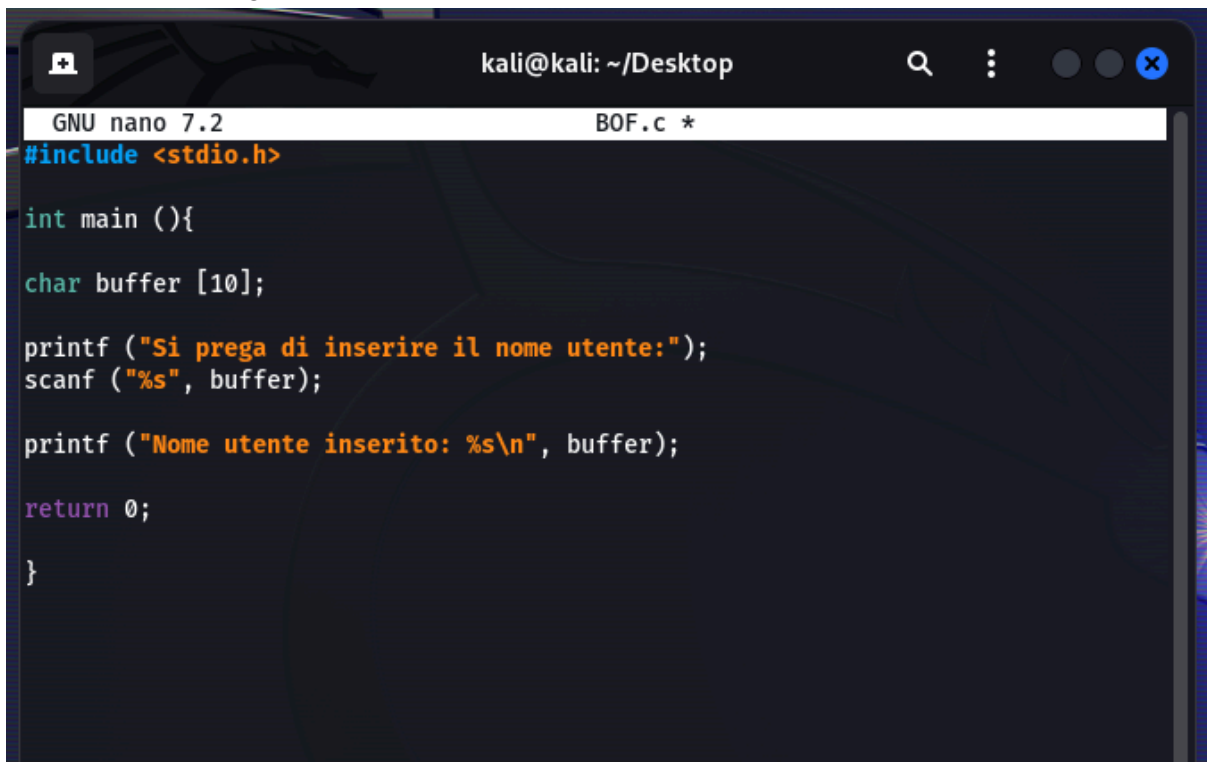
Lo scopo dell'esercizio di oggi è creare un codice in C volutamente ai Buffer OverFlow (vulnerabilità che è conseguenza di una mancanza di controllo dei limiti dei buffer che accettano input utente) che scatena un errore chiamato segmentation fault, ovvero un errore di memoria che si presenta quando un programma cerca inavvertitamente di scrivere su una posizione di memoria dove non gli è permesso scrivere.

Una volta fatto si testa il codice provando l'errore, si modifica il codice in modo che l'errore non si verifichi e si verifica, modificando il codice, dove va a scrivere i caratteri in overflow.

CREAZIONE CODICE IN C VULNERABILE AI BOF

Si apre un terminale di Kali e ci si sposta nella **directory Desktop** con **cd /home/Kali/Desktop** e si crea il file in C con **nano BOF.c**.

Ora inseriamo il seguente codice in C vulnerabile ai BOF e salviamo il file:



```
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 utente inserito: %s\n", buffer);

    return 0;
}
```

A questo punto compiliamo il file utilizzando il comando **gcc -g BOF.c -o BOF** ed eseguiamo il programma con il comando **./BOF**.



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

(kali@kali)-[~/Desktop]
$ ./BOF
```

Testiamo degli inserimenti di **stringhe di caratteri**, sia nei limiti consentiti (inserimento giusto) che oltre i limiti consentiti (inserimento sbagliato, Buffer OverFlow).

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

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

Sistemiamo il codice **aumentando il numero massimo di caratteri consentiti** dal programma durante l'inserimento, aumentando il char, per esempio, a **30**.



```
GNU nano 7.2 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;

}

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

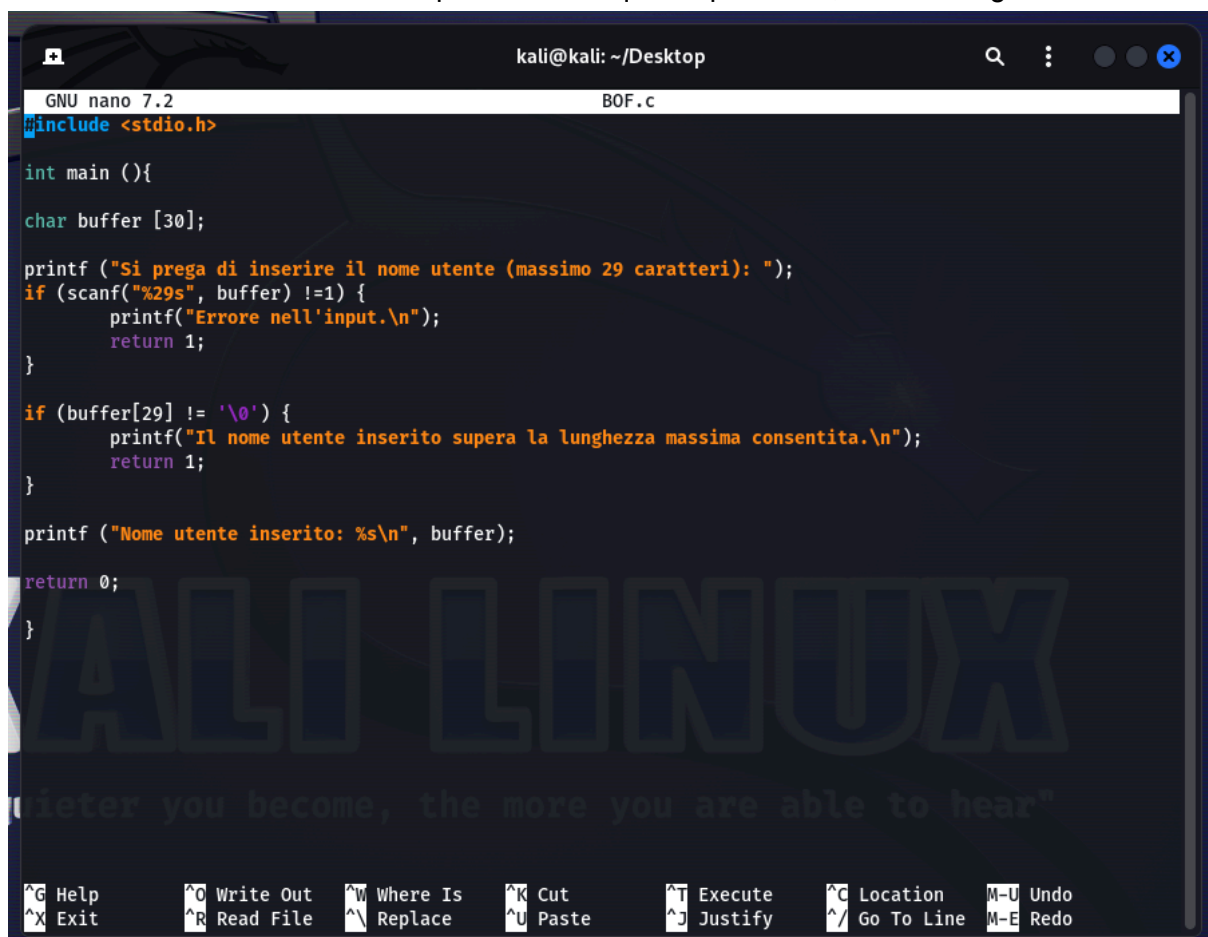
Facciamo delle nuove prove inserendo, come prima, una stringa di caratteri giusta e una sbagliata:

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

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

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

Creiamo un codice con i controlli per ovviare a questo problema, come di seguito:



```
GNU nano 7.2 BOF.c
#include <stdio.h>

int main (){
    char buffer [30];

    printf ("Si prega di inserire il nome utente (massimo 29 caratteri): ");
    if (scanf("%29s", buffer) !=1) {
        printf("Errore nell'input.\n");
        return 1;
    }

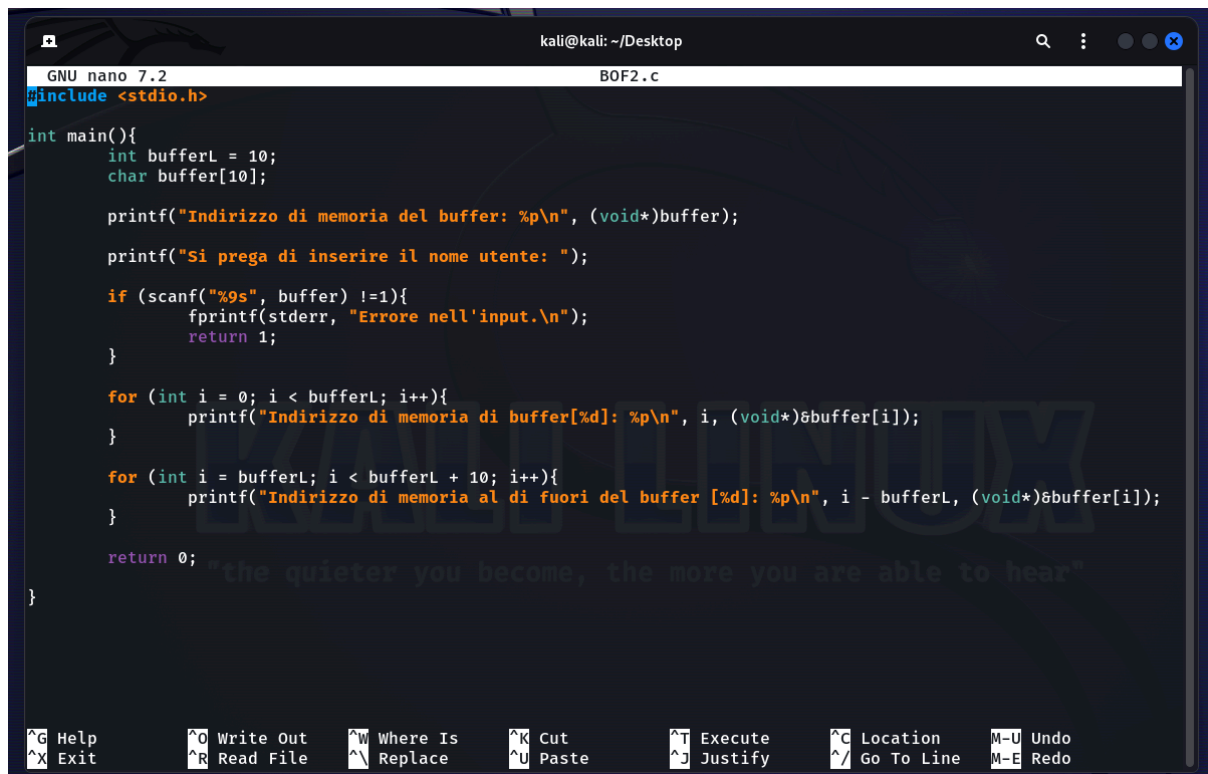
    if (buffer[29] != '\0') {
        printf("Il nome utente inserito supera la lunghezza massima consentita.\n");
        return 1;
    }

    printf ("Nome utente inserito: %s\n", buffer);
    return 0;
}
```

Facciamo nuovamente delle nuove prove inserendo, come prima, una stringa di caratteri giusta e una sbagliata:

A questo punto possiamo creare un nuovo programma che stampi gli indirizzi di memoria del buffer.

Il codice è quello che segue:



```
GNU nano 7.2 BOF2.c
#include <stdio.h>

int main(){
    int bufferL = 10;
    char buffer[10];

    printf("Indirizzo di memoria del buffer: %p\n", (void*)buffer);

    printf("Si prega di inserire il nome utente: ");

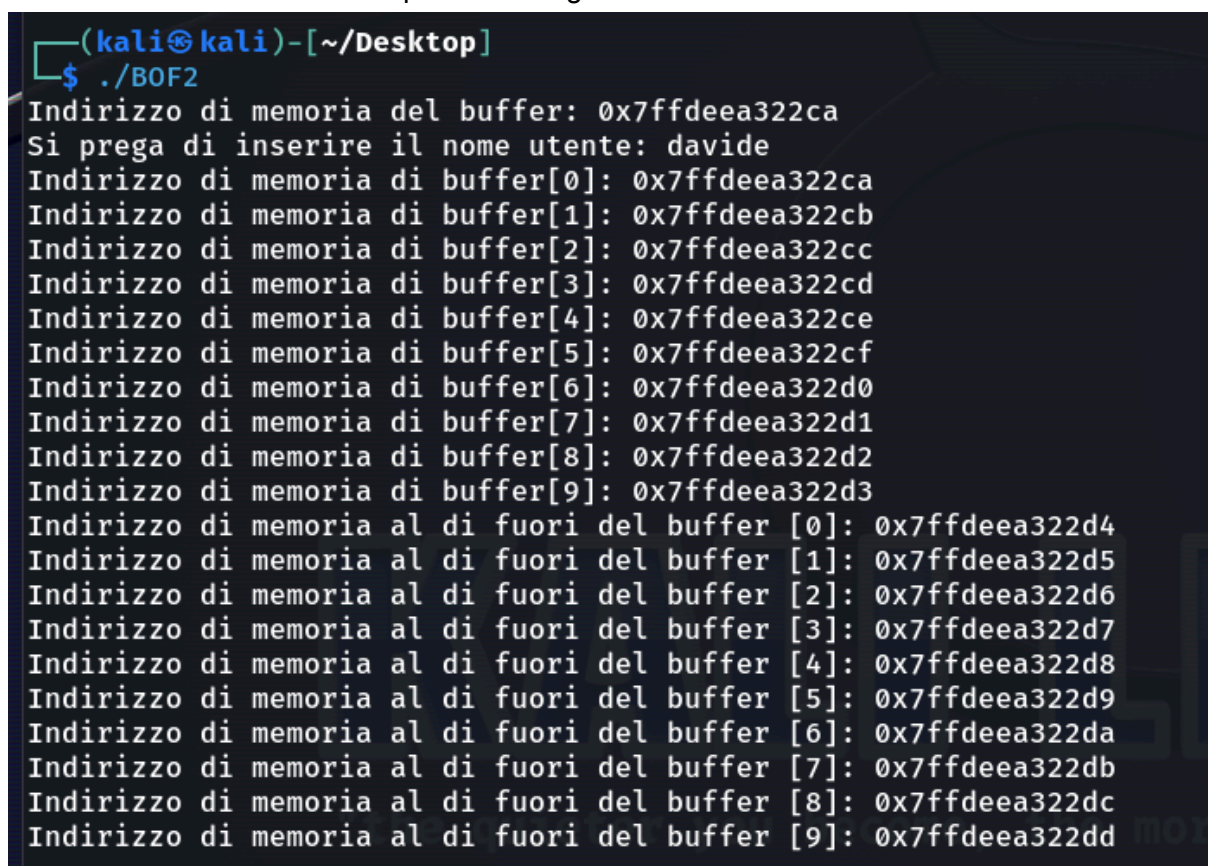
    if (scanf("%9s", buffer) != 1){
        fprintf(stderr, "Errore nell'input.\n");
        return 1;
    }

    for (int i = 0; i < bufferL; i++){
        printf("Indirizzo di memoria di buffer[%d]: %p\n", i, (void*)&buffer[i]);
    }

    for (int i = bufferL; i < bufferL + 10; i++){
        printf("Indirizzo di memoria al di fuori del buffer [%d]: %p\n", i - bufferL, (void*)&buffer[i]);
    }

    return 0;
}
```

Testiamolo e dovrebbe uscire quello che segue:



```
(kali@kali)-[~/Desktop]
$ ./BOF2
Indirizzo di memoria del buffer: 0x7ffdeea322ca
Si prega di inserire il nome utente: dave
Indirizzo di memoria di buffer[0]: 0x7ffdeea322ca
Indirizzo di memoria di buffer[1]: 0x7ffdeea322cb
Indirizzo di memoria di buffer[2]: 0x7ffdeea322cc
Indirizzo di memoria di buffer[3]: 0x7ffdeea322cd
Indirizzo di memoria di buffer[4]: 0x7ffdeea322ce
Indirizzo di memoria di buffer[5]: 0x7ffdeea322cf
Indirizzo di memoria di buffer[6]: 0x7ffdeea322d0
Indirizzo di memoria di buffer[7]: 0x7ffdeea322d1
Indirizzo di memoria di buffer[8]: 0x7ffdeea322d2
Indirizzo di memoria di buffer[9]: 0x7ffdeea322d3
Indirizzo di memoria al di fuori del buffer [0]: 0x7ffdeea322d4
Indirizzo di memoria al di fuori del buffer [1]: 0x7ffdeea322d5
Indirizzo di memoria al di fuori del buffer [2]: 0x7ffdeea322d6
Indirizzo di memoria al di fuori del buffer [3]: 0x7ffdeea322d7
Indirizzo di memoria al di fuori del buffer [4]: 0x7ffdeea322d8
Indirizzo di memoria al di fuori del buffer [5]: 0x7ffdeea322d9
Indirizzo di memoria al di fuori del buffer [6]: 0x7ffdeea322da
Indirizzo di memoria al di fuori del buffer [7]: 0x7ffdeea322db
Indirizzo di memoria al di fuori del buffer [8]: 0x7ffdeea322dc
Indirizzo di memoria al di fuori del buffer [9]: 0x7ffdeea322dd
```