

# Report Buffer overflow W17D4

In questo esercizio andremo a simulare un errore chiamato segmentation fault, per farlo compileremo un codice in C su Kali per l'inserimento di un username.

**Step 1** Sulla macchina Kali andiamo a creare un file di testo .c per compilare il codice indicato dall'esercizio.

Il codice permette l'inserimento di un username di massimo 10 caratteri.

```
GNU nano 8.3 BOF.c *
#include <stdio.h>

int main () {
    char buffer [10];

    printf ("Inserire nome utente ");
    scanf ("%s", buffer);

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

    return 0;
}
```

**Step 2** Compiliamo il codice con `gcc -g <nome file.c> -o <nome programma>`.

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

**Step 3** Mandiamo in esecuzione con `./<nome programma>`.

Se inseriamo un username con meno di 10 caratteri, il programma accetterà l'input normalmente, se invece i caratteri superano la soglia massima, verrà stampato a schermo il messaggio di errore "segmentation fault".

```
(kali㉿kali)-[~/Desktop/Cprograms]
$ ./BOF
Inserire nome utente Zelda
Nome utente inserito: Zelda

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

**Step 4** Provando ad aumentare il parametro dei caratteri che il codice accetta in input (come suggerito dalla traccia dell'esercizio), per verificare se è sufficiente per l'eliminazione dell'errore visto in precedenza.

```
GNU nano 8.3 BOF.c *
```

```
#include <stdio.h>

int main () {

    char buffer [30];

    printf ("Inserire nome utente ");
    scanf ("%s", buffer);

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

    return 0;

}
```

**Step 5** Compiliamo nuovamente il codice come visto in precedenza e mandiamolo in esecuzione, come possiamo vedere basterà aggiungere un numero di caratteri superiori a 30 perché l'errore si verifichi nuovamente.

Non è quindi sufficiente aumentare il numero di caratteri che il codice può accettare in input per mitigare l'errore.

[illegible]

## Facoltativo

Integrare nel programma esistente delle verifiche di sicurezza per prevenire buffer overflow (BOF) e procedure di sanitizzazione dell'input.

Modifiche apportate al codice:

```
GNU nano 8.3 BOF.c *
#include <stdio.h>

int main() {
    char b[11];
    int i;

    printf("Inserire nome utente (max 10 caratteri): ");
    if (fgets(b, sizeof(b), stdin)) {
        for (i = 0; i < 11; i++) {
            if (b[i] == '\n') {
                b[i] = 0;
                printf("Nome utente accettato: %s\n", b);
                return 0;
            }
        }
    }
    printf("Errore: input troppo lungo o lettura fallita.\n");
    return 1;
}
```

Output del codice se viene inserito un nome utente con meno di 10 caratteri e con più di 10 caratteri:

```
(kali@kali)-[~/Desktop/Cprograms]
$ ./BOF
Inserire nome utente (max 10 caratteri): Sara
Nome utente accettato: Sara

(kali@kali)-[~/Desktop/Cprograms]
$ ./BOF
Inserire nome utente (max 10 caratteri): Sarararararara
Errore: input troppo lungo o lettura fallita.
```