



Emanuele Di Stefano

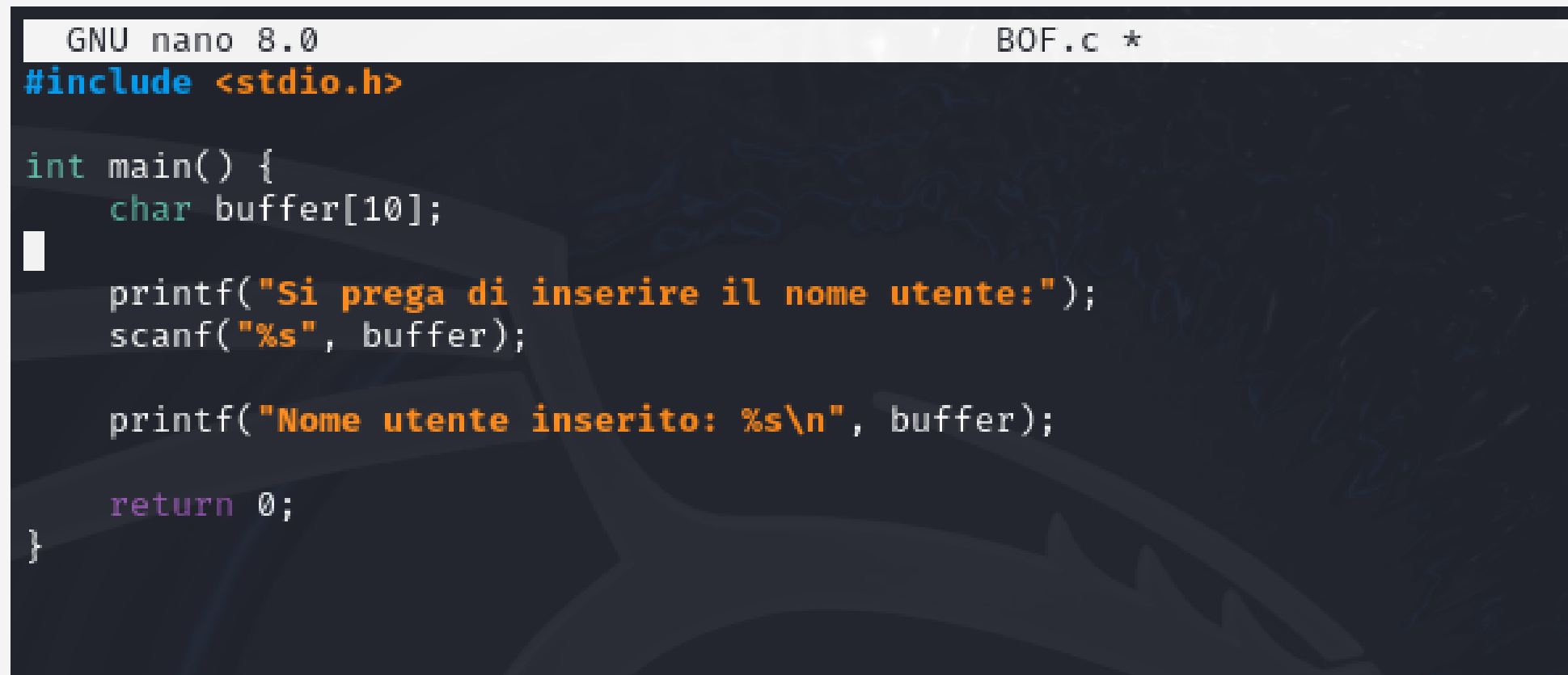
# Relazione su Buffer Overflow in C

S7-L4



## Screenshot 1: Introduzione

Il buffer overflow si verifica quando vengono scritti più dati in un buffer rispetto alla sua capacità, causando la sovrascrittura della memoria adiacente. Questo fenomeno può portare a comportamenti imprevisti del programma, crash del sistema e vulnerabilità di sicurezza.



```
GNU nano 8.0 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;
}
```

### Descrizione del Codice:

**Dichiarazione del Buffer:** Il programma dichiara un array di caratteri chiamato buffer con una dimensione di 10 byte.

**Input dell'Utente:** Viene letta una stringa di caratteri dall'input standard (di solito la tastiera) e memorizzata nel buffer. Tuttavia, scanf non verifica la lunghezza della stringa inserita.

**Stampa della Stringa:** Il programma stampa la stringa memorizzata nel buffer.

## Fasi del Test:

Creazione del file di codice:

Utilizzo dell'editor di testo nano per scrivere il codice.

Comando: nano BOF.c

Compilazione del codice:

Utilizzo di gcc per compilare il codice.

Comando: gcc -g BOF.c -o BOF

Esecuzione del programma:

Comando: ./BOF

Inserimento di un nome utente lungo per testare il comportamento del programma con input eccessivo.

```
lelo@lelo: ~/Desktop
File Actions Edit View Help
(lelo@lelo)-[~]
$ cd Desktop/

(lelo@lelo)-[~/Desktop]
$ nano BOF.C

(lelo@lelo)-[~/Desktop]
$ gcc -g BOF.c -o BOF
cc1: fatal error: BOF.c: No such file or directory
compilation terminated.

(lelo@lelo)-[~/Desktop]
$ ls
BOF.C      'SCAN_nessus'  file_php      starting_point_Lelo34.ovpn
Python    Thema_Sweet    prova_meta    use

(lelo@lelo)-[~/Desktop]
$ gcc -g BOF.c -o BOF
cc1: fatal error: BOF.c: No such file or directory
compilation terminated.

(lelo@lelo)-[~/Desktop]
$ ./BOF
bash: ./BOF: No such file or directory

(lelo@lelo)-[~/Desktop]
$ rm BOF.C

(lelo@lelo)-[~/Desktop]
$ nano BOF.c

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

(lelo@lelo)-[~/Desktop]
$ ./BOF
Si prega di inserire il nome utente:trololololololololololololololololololololololololol
Nome utente inserito: trololololololololololololololololololololololololol
Segmentation fault

(lelo@lelo)-[~/Desktop]
$
```

Modificare il Codice per un Buffer più Grande  
Per risolvere il problema, possiamo aumentare la dimensione del buffer.

Apriamo nuovamente il file BOF.c con nano:

```
telo@telo: ~/Desktop
File Actions Edit View Help
GNU nano 8.0 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;
}
```

## Compilare e Eseguire di Nuovo

Compiliamo nuovamente il codice con il comando:

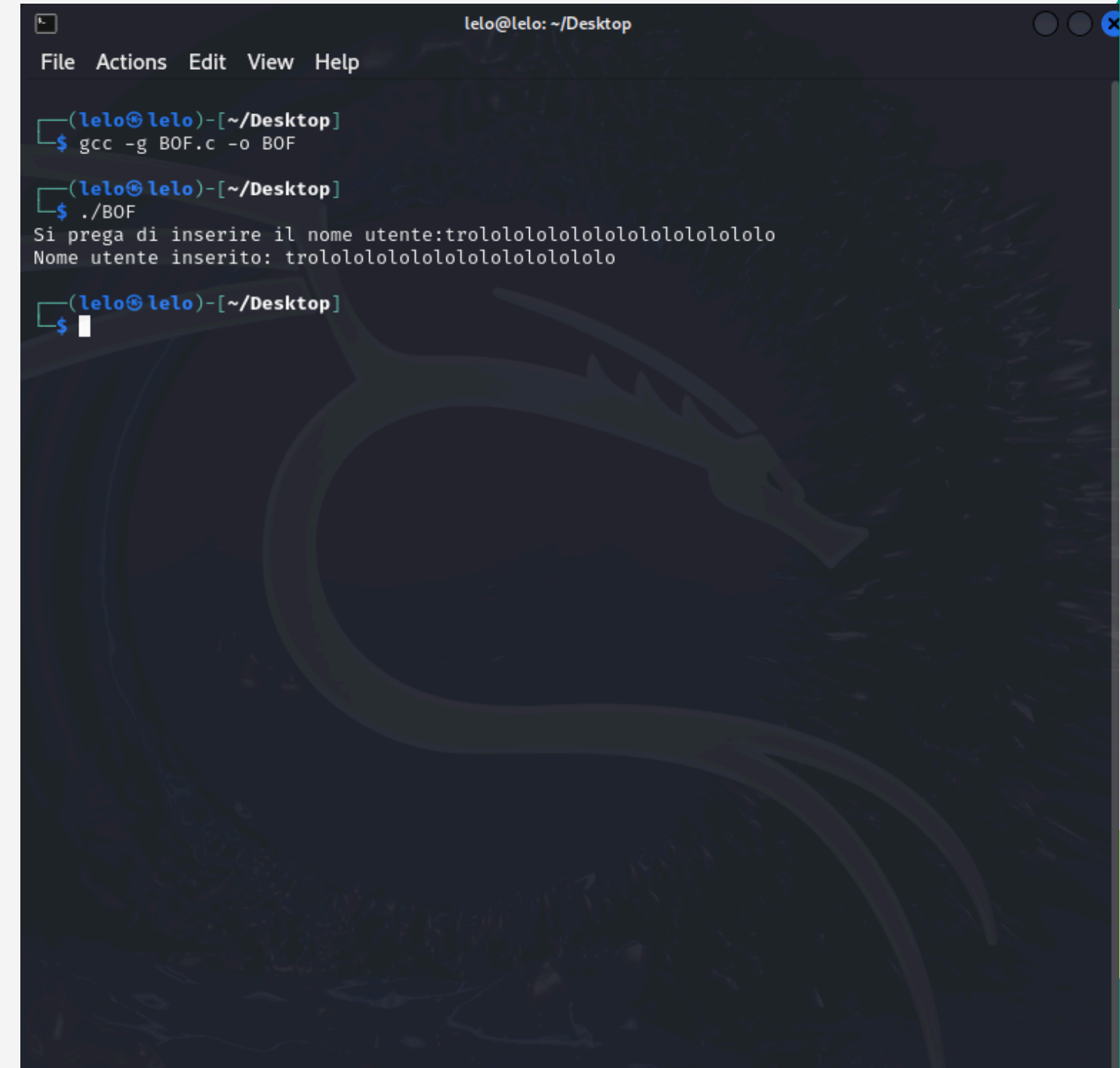
```
gcc -g BOF.c -o BOF
```

Eseguiamo nuovamente il programma per verificare che ora accetti input più lunghi senza errori di segmentazione:

```
./BOF
```

## Testare il Programma

Infine, inseriamo un nome utente lungo e verifichiamo che il programma non restituisca un errore di segmentazione. Dovremmo vedere il nome utente completo stampato correttamente.



```
lelo@lelo: ~/Desktop
File Actions Edit View Help

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

(lelo@lelo)~[/Desktop]
$ ./BOF
Si prega di inserire il nome utente:trolololololololololololololo
Nome utente inserito: trololololololololololololololo

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

## Prove per misurare il crash:

Qui ho fatto dei test per dimostrare la misura precisa per ottenere l'overflow, in questo caso risulta che con vettore 10 = a 18 per ottenere il segmentation fault.

Mentre a vettore 30 = a 41 per ottenerlo.

[illegible]



## **Prevenzione del Buffer Overflow**

Prevenzione del Buffer Overflow:

**Utilizzare scanf con specificatore di lunghezza:**

```
-scanf("%9s", buffer);
```

**Utilizzare fgets:**

```
fgets(buffer, sizeof(buffer), stdin);
```

**Aumento della Dimensione del Buffer:**

Modifica della dimensione del buffer a 30 byte per evitare overflow con input più lunghi

