



S7/L4

BUFFER OVERFLOW

FLAVIO SCOGNAMIGLIO



BUFFER OVERFLOW

La traccia di oggi ci invita ad un'analisi guidata al fine di toccare con mano il buffer overflow. Un buffer overflow è una vulnerabilità di sicurezza che si verifica quando un programma scrive **più dati in un buffer di quanto lo stesso sia in grado di contenere**. I buffer sono aree di memoria allocate per contenere dati temporanei, come stringhe di input utente. In C, un buffer overflow si verifica tipicamente quando non c'è un controllo adeguato sui limiti dell'input. Di seguito vediamo l'esempio di un codice volutamente **vulnerabile** proposto dall'esercitazione.

The image shows three overlapping Parrot Terminal windows. The leftmost window shows the user navigating to the Desktop and creating a file named B0F.c using nano. The middle window shows the contents of B0F.c, which is a C program with a 10-character buffer. The rightmost window shows the compilation of B0F.c into an executable named B0F, followed by its execution, which prompts the user for a name. A red arrow points from the text on the left to the execution window.

```
[flavio@parrot]-[~]  
$cd Desktop/  
[flavio@parrot]-[~/Desktop]  
$nano B0F.c
```

```
GNU nano 7.2 B0F.c Modificato  
#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;  
}
```

```
[flavio@parrot]-[~/Desktop]  
$gcc -g B0F.c -o B0F  
[flavio@parrot]-[~/Desktop]  
$./B0F  
Si prega di inserire il nome utente:
```

[riga 15/15 (100%), colonna 1/ 1 (100%), carattere 188/188 (100%)]
^H Guida ^O Inserisci ^R Sostituisci ^V Incolla ^G Vai a riga ^Y Ripeti
^X Esci ^F Cerca ^K Taglia ^T Esegui ^Z Annulla M-A Set Mark

Creo un file in C con un array
che può contenere al massimo
10 caratteri. Lo compilo e lo
eseguo

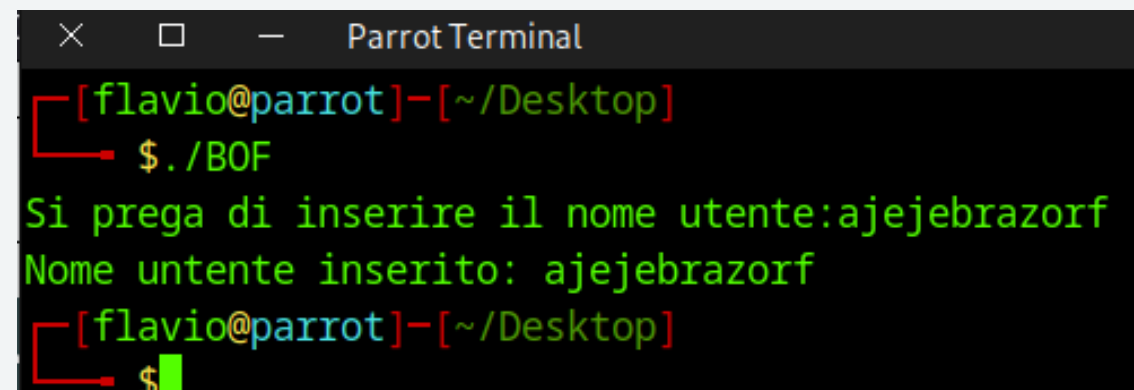
BUFFER OVERFLOW [10]

In questo esempio, buffer è un array di 10 caratteri. La funzione `scanf("%s", buffer)` legge una stringa di input dall'utente e la memorizza in buffer. Tuttavia, `scanf` non limita il numero di caratteri letti, quindi se l'utente inserisce più di 9 caratteri (più il terminatore nullo), i dati eccedenti **sovrascriveranno la memoria adiacente**.

Nello specifico, invece, un **segmentation fault** è un tipo di **errore di memoria** che si verifica quando un programma tenta di accedere a una porzione di memoria a cui **non è autorizzato ad accedere**. Questo errore è spesso il risultato di un buffer overflow, dove la memoria al di fuori dei limiti del buffer viene modificata, causando corruzione della memoria e, eventualmente, un tentativo di accesso non valido.

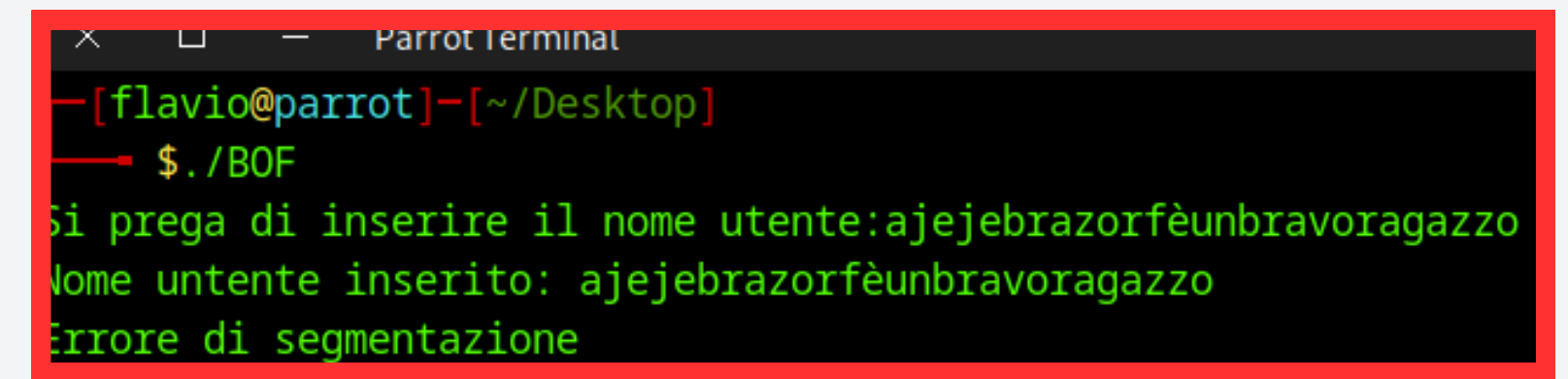
TEST

In ambiente Linux, nonostante l'inserimento di più di 10 caratteri, il programma continua a funzionare senza errori visibili. Tuttavia, la memoria adiacente viene corrotta.



```
Parrot Terminal
[flavio@parrot]~[/Desktop]
$ ./BOF
Si prega di inserire il nome utente: ajejebrazorf
Nome utente inserito: ajejebrazorf
[flavio@parrot]~[/Desktop]
$
```

Inserendo un input più lungo, il programma tenta di accedere a memoria **non autorizzata**, causando un segmentation fault a causa del buffer overflow.

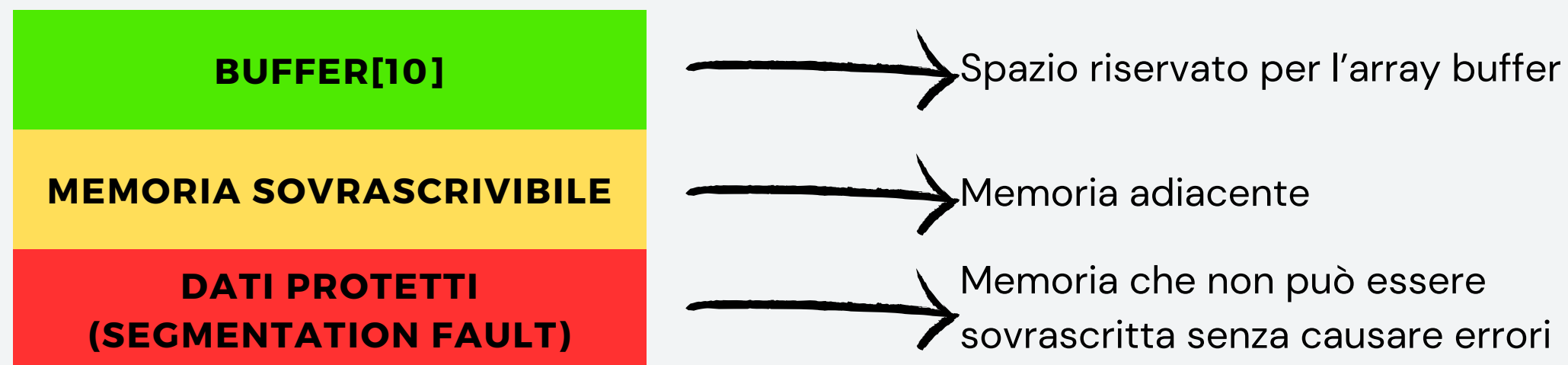


```
Parrot Terminal
[flavio@parrot]~[/Desktop]
$ ./BOF
Si prega di inserire il nome utente: ajejebrazorfèunbravoragazzo
Nome utente inserito: ajejebrazorfèunbravoragazzo
Errore di segmentazione
```

PERCHÉ IL SEGMENTATION FAULT AVVIENE DOPO 18 CARATTERI?

In questo specifico caso (LINUX), dopo un tot di caratteri si verifica il segmentation fault. Quando si scrive oltre i limiti di buffer, si entra in una zona di memoria che non è stata allocata per il buffer stesso. Il comportamento di un programma che accede alla memoria oltre i limiti dell'array è **indefinito** e dipende dall'**organizzazione della memoria del programma**.

Nel caso specifico del codice fornito dalla traccia, il segmentation fault potrebbe avvenire dopo un tot di caratteri perché i primi 8 byte oltre il buffer potrebbero corrispondere a una parte della memoria che non causa immediatamente un errore, come altre **variabili locali** o **padding** del compilatore. Tuttavia, continuando a sovrascrivere memoria, si arriva a una zona che è **protetta dal sistema operativo**, quindi **inaccessibile**, come una porzione di stack non allocata o memoria riservata per altre funzionalità del programma. Quando questa area viene toccata, il sistema operativo **genera un segmentation fault per proteggere la memoria**.

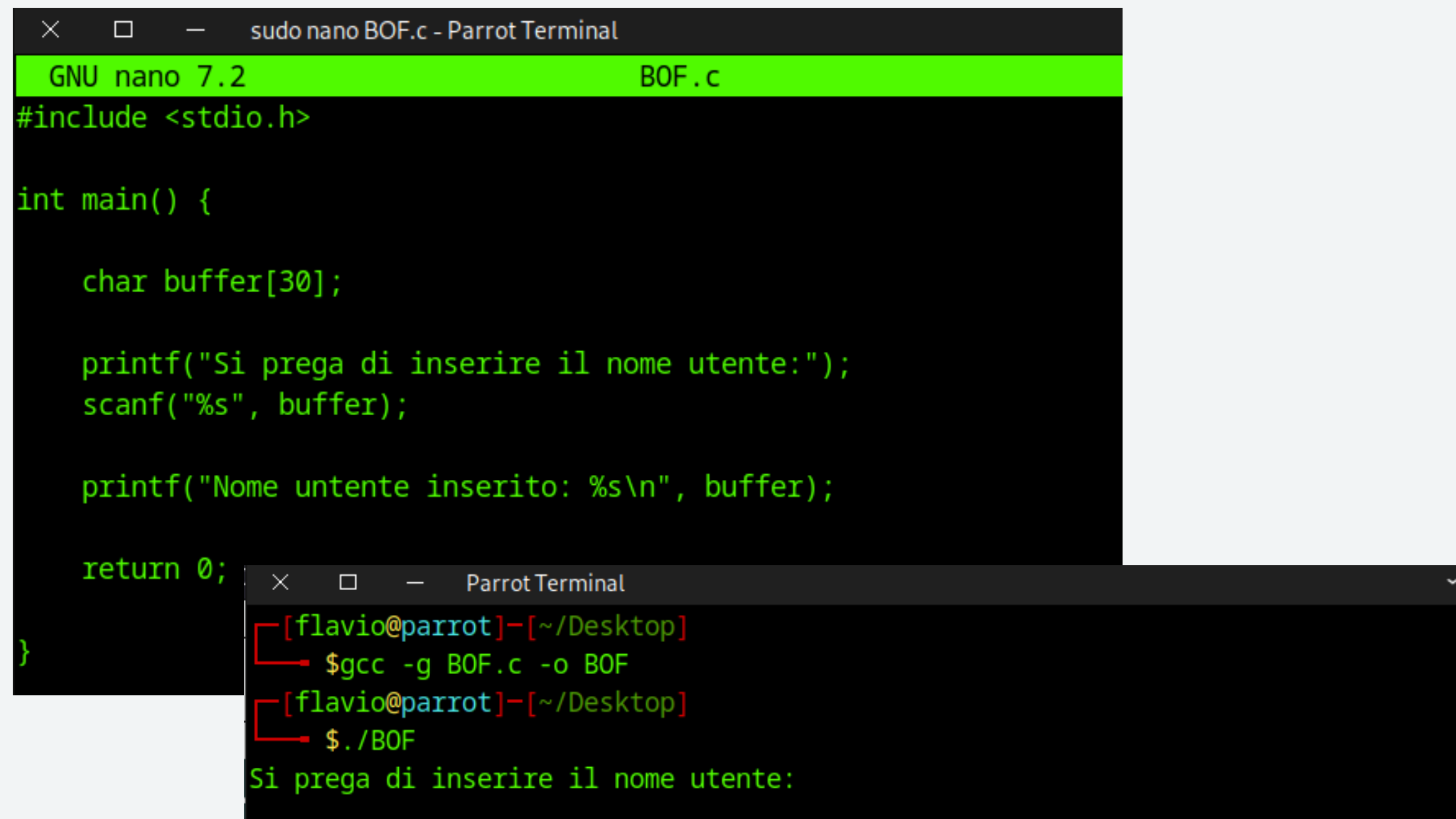


Su MacOS ad esempio, lo stesso input dato in pasto al programma mostra subito un alert dalla shell zsh

```
flavio@MacBook-Pro Desktop % ./test
Si prega di inserire il nome utente: ajejebrazorf
Nome utente inserito: ajejebrazorf
zsh: abort ./test
flavio@MacBook-Pro Desktop %
```

BUFFER OVERFLOW [30]

A questo punto, come da richiesta, **modifichiamo** il programma affinché accetti 30 caratteri anziché 10. **Compiliamo nuovamente il programma, avendo effettuato modifiche, ed eseguiamolo.**



The image shows two overlapping terminal windows. The top window, titled 'sudo nano BOF.c - Parrot Terminal', displays the source code of a C program in the nano editor. The code defines a character array 'buffer' of size 30 and prompts the user to enter a name. The bottom window, titled 'Parrot Terminal', shows the compilation and execution of the program. It displays the command prompt for 'flavio@parrot' in the directory '~/Desktop', followed by the compilation command '\$gcc -g BOF.c -o BOF', the execution command '\$./BOF', and the program's output 'Si prega di inserire il nome utente:'.

```
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;
}

[flavio@parrot]~$ gcc -g BOF.c -o BOF
[flavio@parrot]~$ ./BOF
Si prega di inserire il nome utente:
```

BUFFER OVERFLOW [30]

A questo punto, come da richiesta, **modifichiamo** il programma affinché accetti 30 caratteri anziché 10. **Compiliamo nuovamente il programma, avendo effettuato modifiche, ed eseguiamolo.**

Dato che il buffer questa volta accetta 30 caratteri, non vi è alcun problema. Nel momento in cui eccediamo e inseriamo più di 30 caratteri si verifica l'overflow del buffer, dove il programma tenta di scrivere contenuti su una porzione di memoria alla quale non ha accesso.

```
flavio$ ./BOF
Si prega di inserire il nome utente: ajejebrazorf1234567890qwerty123456789qwertyaz
ccx
Nome utente inserito: ajejebrazorf1234567890qwerty123456789qwertyazccx
zsh: segmentation fault ./BOF
```

```
flavio@MacBook-Pro Desktop % ./test
Si prega di inserire il nome utente: ajejebrazorf123456789
Nome utente inserito: ajejebrazorf123456789
flavio@MacBook-Pro Desktop % ./test
Si prega di inserire il nome utente: ajejebrazorf123456789azxcvdfderwqs
Nome utente inserito: ajejebrazorf123456789azxcvdfderwqs
zsh: abort ./test
flavio@MacBook-Pro Desktop %
```

GRAZIE