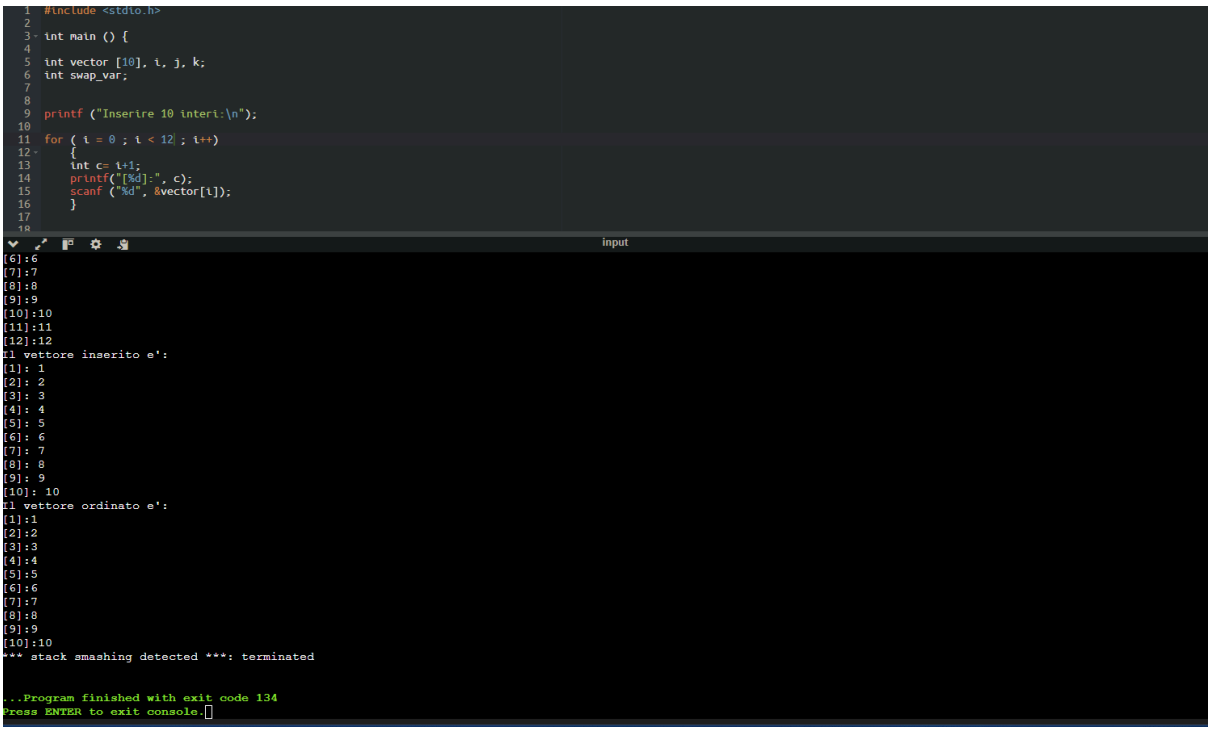
Buffer Overflow

Un buffer overflow si verifica quando un programma scrive più dati di quanti un buffer (un'area di memoria) può contenere. In termini tecnici, il programma accede a regioni di memoria adiacenti, sovrascrivendo dati non previsti e potenzialmente compromettendo la stabilità del software o la sicurezza del sistema. In modo metaforico, immagina di versare acqua in una bottiglia fino a quando trabocca: il liquido in eccesso può causare problemi all'ambiente circostante, proprio come dati in eccesso possono danneggiare un programma o esporlo ad attacchi informatici. Il buffer overflow è un noto vettore d'attacco che può permettere a un malintenzionato di eseguire codice arbitrario o di accedere a informazioni sensibili.

```
#include <stdio.h>

int main () {

int vector [10], i, j, k;
int swap_var;


printf ("Inserire 10 interi:\n");

for ( i = 0 ; i < 10 ; i++)
        {
        int c= i+1;
        printf("[%d]:", c);
        scanf ("%d", &vector[i]);
        }

printf ("Il vettore inserito e':\n");
for ( i = 0 ; i < 10 ; i++)
        {
        int t= i+1;
        printf("[%d]: %d", t, vector[i]);
        printf("\n");
        }

for (j = 0 ; j < 10 - 1; j++)
        {
        for (k = 0 ; k < 10 - j - 1; k++)
                {
                        if (vector[k] > vector[k+1])
                        {
                        swap_var=vector[k];
                        vector[k]=vector[k+1];
                        vector[k+1]=swap_var;
                        }
                }
        }
printf("Il vettore ordinato e':\n");
for (j = 0; j < 10; j++)
        {
        int g = j+1;
        printf("[%d]:", g);
        printf("%d\n", vector[j]);
        }

return 0;

}
```

Il codice originale era un programma in C progettato per ordinare un array di numeri interi utilizzando l'algoritmo del bubble sort. L'utente inserisce dieci numeri, che vengono poi ordinati e stampati in ordine crescente. Il programma inizia dichiarando un array vector di dieci elementi e tre variabili intere i, j e k per i cicli di iterazione. La funzione printf fornisce istruzioni all'utente per inserire i numeri, e scanf li raccoglie nell'array. Dopo aver raccolto i numeri, vengono stampati per mostrare il vettore non ordinato. Infine, il programma applica il bubble sort ai dieci valori e li stampa in ordine. Testandolo velocemente in un compiler online, abbiamo potuto confermare le nostre ipotesi.



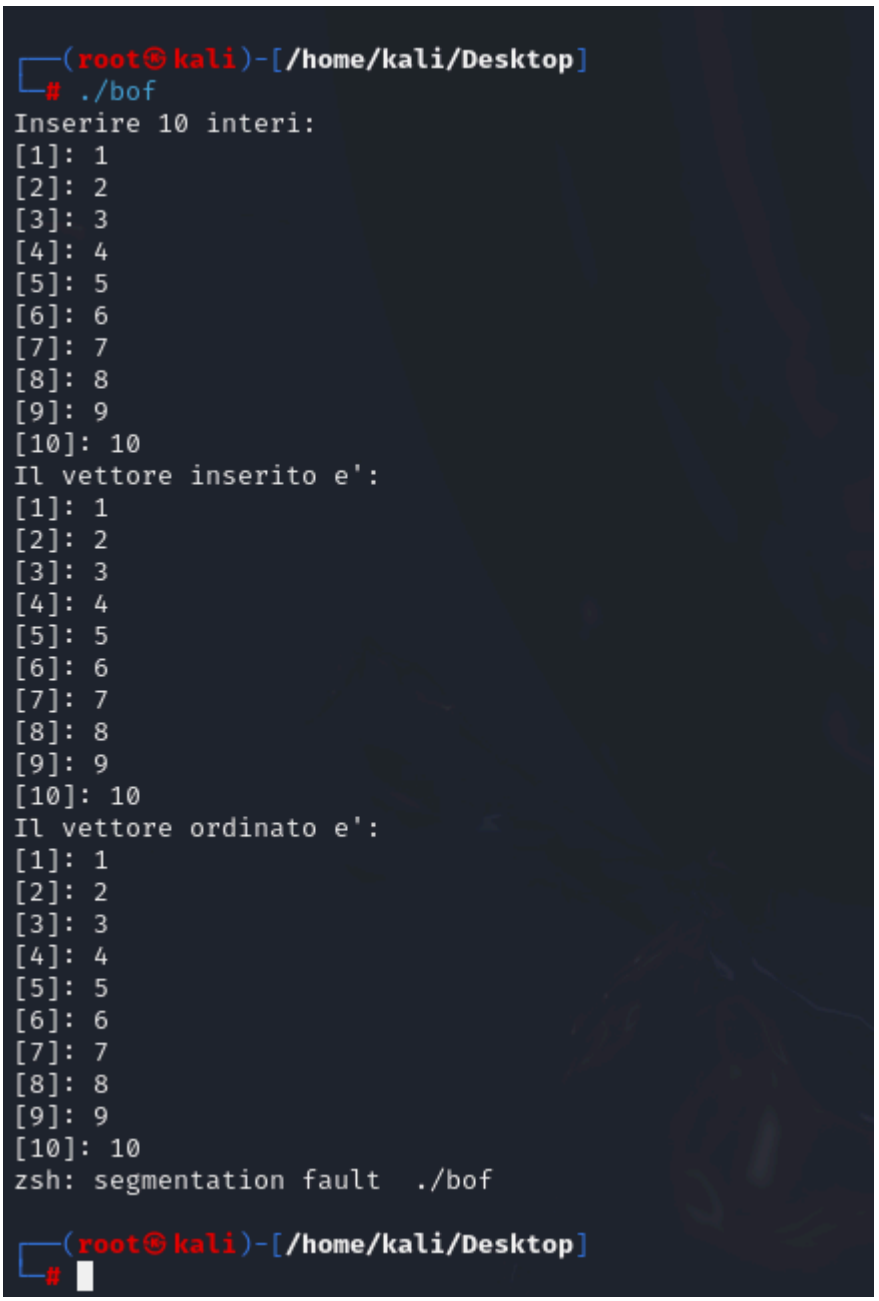
```c
#include <stdio.h>

int main () {

int vector [10], i, j, k;
int swap_var;


printf ("Inserire 10 interi:\n");

for ( i = 0 ; i < 12 ; i++)
    {
    int c= i+1;
    printf("[%d]:", c);
    scanf ("%d", &vector[i]);
    }
```

```
input
[6]:6
[7]:7
[8]:8
[9]:9
[10]:10
[11]:11
[12]:12
Il vettore inserito e':
[1]: 1
[2]: 2
[3]: 3
[4]: 4
[5]: 5
[6]: 6
[7]: 7
[8]: 8
[9]: 9
[10]: 10
Il vettore ordinato e':
[1]:1
[2]:2
[3]:3
[4]:4
[5]:5
[6]:6
[7]:7
[8]:8
[9]:9
[10]:10
*** stack smashing detected ***: terminated


...Program finished with exit code 134
Press ENTER to exit console.
```

Nel primo tentativo di provocare un buffer overflow, abbiamo modificato il ciclo di input utente per accettare 12 valori invece dei 10 previsti. Questa modifica ha portato il programma a sovrascrivere la memoria al di fuori dei limiti dell'array vector[10]. Testandolo innanzitutto in un compiler online, possiamo subito registrare che il programma è stato interrotto perché è stato rilevato stack smashing; un errore di stack smashing si verifica quando il sistema rileva che il codice, uscendo dai limiti della memoria allocata per un array, sta tentando di sovrascrivere dati critici nello stack, come indirizzi di ritorno o altre informazioni importanti. Possiamo quindi confermare di aver effettuato un buffer overflow con successo e che il nostro .

File Edit Search View Document Help

```c
#include <stdio.h>

int main() {
    int vector[10], i, j, k;
    int swap_var;

    // Inserimento dei primi 10 numeri da parte dell'utente
    printf("Inserire 10 interi:\n");
    for (i = 0; i < 10; i++) {
        printf("[%d]: ", i + 1);
        scanf("%d", &vector[i]);
    }

    // Riempimento automatico di altri 200 numeri nel buffer (sovrascrivendo)
    for (i = 10; i < 210; i++) {  // Dal 10° al 210° elemento
        vector[i] = i;  // Assegna valori sequenziali o un pattern desiderato
    }

    // Stampa del vettore inserito
    printf("Il vettore inserito e':\n");
    for (i = 0; i < 10; i++) {
        printf("[%d]: %d\n", i + 1, vector[i]);
    }

    // Ordinamento del vettore (solo i primi 10 valori per il bubble sort)
    for (j = 0; j < 10 - 1; j++) {
        for (k = 0; k < 10 - j - 1; k++) {
            if (vector[k] > vector[k + 1]) {
                swap_var = vector[k];
                vector[k] = vector[k + 1];
                vector[k + 1] = swap_var;
            }
        }
    }

    // Stampa del vettore ordinato (solo i primi 10 valori)
    printf("Il vettore ordinato e':\n");
    for (j = 0; j < 10; j++) {
        printf("[%d]: %d\n", j + 1, vector[j]);
    }

    return 0;
}
```



Nel secondo screenshot, per amplificare l’impatto del buffer overflow, abbiamo aggiunto un ciclo che inserisce 200 valori aggiuntivi nel buffer, scrivendo dati fuori dai limiti dell’array originale. L'array vector è dichiarato con una dimensione di 10 elementi, ma il ciclo estende l'array fino al 210° elemento, violando chiaramente la memoria. Questo overflow può causare un errore di segmentazione, che si verifica quando un programma tenta di accedere a una posizione di memoria non autorizzata. Tali errori spesso portano a crash del programma, ma in alcuni casi, possono essere sfruttati per eseguire codice dannoso. Per favorire questo tipo di exploit, sono state disabilitate alcune protezioni di sicurezza tramite comandi GCC come -fno-stack-protector (per disabilitare la protezione dello stack) e -z execstack (per permettere l'esecuzione di codice nella stack).

Termini utili:

Buffer Overflow: Un errore che si verifica quando un programma scrive più dati di quanti ne siano stati allocati in un buffer, corrompendo la memoria adiacente.

Stack: La parte della memoria dove vengono gestite variabili locali e informazioni di controllo, come indirizzi di ritorno delle funzioni.

Array: Una struttura dati che contiene un numero fisso di elementi dello stesso tipo, accessibili tramite indice.

Stack Smashing: Un tipo di buffer overflow in cui dati non controllati sovrascrivono lo stack, minacciando la sicurezza del sistema.

Errore di Segmentazione: Un errore generato quando un programma tenta di accedere a memoria non permessa o fuori dai limiti allocati..

Sovrascrittura: Processo in cui dati eccedenti riscrivono informazioni già allocate causando comportamenti imprevisti.