

# REPORT

## TRACCIA GIORNO 3

Obiettivo:

L'obiettivo dell'esercizio datoci è:

- Descrivere il funzionamento del programma prima dell'esecuzione;
- Riprodurre ed eseguire il programma;
- Modificare affinché si verifichi un errore di segmentazione.

Il codice che ci è stato dato è il seguente:

```
#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;

}
```

## ANALISI DEL CODICE

Dopo un'attenta analisi del codice, abbiamo detto che fosse un codice in linguaggio C, il quale, chiede all'utente di inserire 10 numeri interi che verranno poi memorizzati dentro un array chiamato **vector**. La prima stampa dell'array, permette all'utente di inserire i numeri, mentre la seconda, visualizza tutti i numeri dall'utente. Infine viene stampato il vettore in ordine crescente.

Una volta eseguito il codice, possiamo dire che le ipotesi sul suo funzionamento fossero corrette, unica cosa da aggiungere per quanto riguarda il riordinamento dei numeri del vettore in ordine crescente, che

ciò viene fatto grazie all'algoritmo **Bubble Sort**. Un algoritmo che scambia e confronta ripetutamente elementi di un array fino a che l'intera sequenza non è ordinata.

## MODIFICA DEL CODICE CON ERRORE DI SEGMENTAZIONE

Il terzo punto della traccia chiedeva di modificare il codice affinché avvenga un'errore di segmentazione, ovvero un'errore che si crea quando il programma, nel nostro caso il vettore, prova ad accedere ad un'area di memoria non consentita, molte volte l'errore di segmentazione si verifica anche quando si tenta di accedere alla memoria con metodi non consentiti.

```
#include <stdio.h>

int main () {

    int vector[10], i, j, k, ntemp, scelta;
    int swap_var;
    i=0;
    j=0;
    k=0;

    while(1){
        printf("----- MENU -----\\n");
        printf("1) Programma corretto\\n");
        printf("2) Programma con possibile BOF\\n");
        printf("-----\\n");
        printf("Scelta: ");
        scanf("%d", &scelta);
        if(scelta > 2){
            printf("Inserisci un valore ammesso.\\n\\n");
        }else{
            break;
        }
    }
}
```

Per la corretta esecuzione dell'esercizio, abbiamo deciso di creare un **Menù**, il quale permetterà all'utente di scegliere tra il **case 1**, che ci eseguirà il codice corretto senza errori di segmentazioni, che è lo stesso codice che ci è stato dato nella traccia dell'esercizio. Quindi andremo direttamente al **case 2**, dato che abbiamo anche cambiato qualcosa nel codice.

## CASE 2

```
printf("\n\n----- SCELTA 2 ----- \n");
printf("Inserire 10 interi:\n");

while(1){
    printf("[%d]", i+1);
    printf("Inserisci il numero. -1 per uscire: ");
    scanf("%d", &ntemp);
    if(ntemp >= 0){
        vector[i] = ntemp;
        i=i+1;
    }else{
        break;
    }
    printf("i= %d, j= %d, k= %d \n",i,j,k);
}

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]);
}

break;

return 0;
}
```

L'opzione 2 è stata pensata per poter generare un errore di segmentation fault causato dal tentativo, da parte dell'applicazione, di scrivere dati in una porzione di memoria a cui non può accedere.

La parte di codice dove è possibile generare il buffer overflow risiede nel primo ciclo while, impiegato per memorizzare l'input utente nell'array dedicato, capace di contenere 10 elementi.

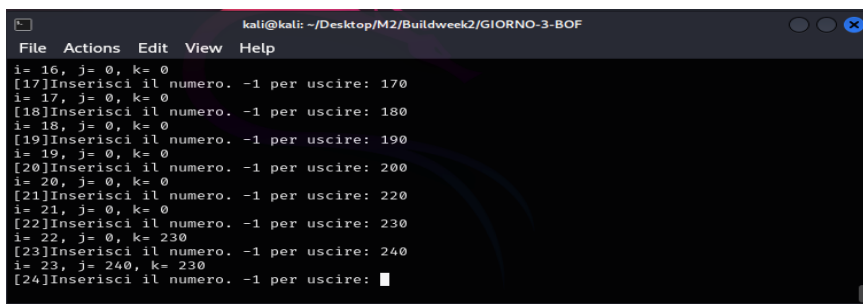
Nel ciclo while, la cui condizione d'iterazione è sempre vera, si chiedono all'utente i numeri da inserire nell'array, specificando che con la digitazione del numero "-1" si procederà ad eseguire il resto del codice.

Come si evince dallo screenshot non è presente alcun controllo sulla quantità di dati inseriti, incrementando, ad ogni ciclo, la variabile "i" che rappresenta la posizione dell'array in cui scrivere.

Proprio questo comportamento del programma da luogo a probabili buffer overflow, in quanto l'utente può liberamente inserire più di 10 elementi nell'array andando, di conseguenza, a scrivere in aree di memoria esterne a quella dedicata a quest'ultimo.

Un esempio è dato dal seguente screenshot dove, alla 22esima iterazione, il programma scrive nello spazio di memoria dedicato alla variabile "k".

Per generare l'errore di segmentazione abbiamo dovuto raggiungere, implementando un inserimento automatico, il 1516esimo inserimento.



```
kali@kali: ~/Desktop/M2/Buildweek2/GIORNO-3-BOF
File Actions Edit View Help
i= 16, j= 0, k= 0
[17]Inserisci il numero. -1 per uscire: 170
i= 17, j= 0, k= 0
[18]Inserisci il numero. -1 per uscire: 180
i= 18, j= 0, k= 0
[19]Inserisci il numero. -1 per uscire: 190
i= 19, j= 0, k= 0
[20]Inserisci il numero. -1 per uscire: 200
i= 20, j= 0, k= 0
[21]Inserisci il numero. -1 per uscire: 220
i= 21, j= 0, k= 0
[22]Inserisci il numero. -1 per uscire: 230
i= 22, j= 0, k= 230
[23]Inserisci il numero. -1 per uscire: 240
i= 23, j= 240, k= 230
[24]Inserisci il numero. -1 per uscire: █
```

## CONCLUSIONI

In conclusione siamo riusciti a raggiungere tutti gli obiettivi richiesti dall'esercizio, riuscendo a capire cosa facesse il codice ancora prima di eseguirlo.

Siamo riusciti a modificare il codice mettendo di fronte all'utente la possibilità di scegliere tra un programma corretto e uno che presenta un'errore di segmentazione.