System Exploit BOF Traccia Giorno 3

1.Nell'esercizio in questione andremo ad analizzare uno codice in C, cercando di capire il suo funzionamento prima di eseguirlo.

```
#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;


}
```

Ecco il codice in questione

1.1Il programma inizia chiedendo all'utente di inserire 10 numeri interi. Questi valori vengono memorizzati all'interno di un vettore, ovvero una struttura dati che permette di raccogliere e gestire più elementi dello stesso tipo.

2.1Una volta terminato l'inserimento, il programma visualizza i numeri così come sono stati forniti. In questa fase, l'ordine dei valori riflette esattamente quello in cui l'utente li ha digitati, senza alcuna modifica o elaborazione.

3.1Successivamente, il programma li ordina in maniera crescente, per poi stamparli. Questo significa che i valori più piccoli si sposteranno all'inizio del vettore, mentre quelli più grandi si troveranno alla fine.

Ora vediamo se le nostre ipotesi sono corrette:

1.Il programma fornito prende in input i numeri dall'utente, salvandoli in un vettore.

```
┌──(kali㉿kali)-[~/Desktop/c++]
└─$ ./traccia
Inserire 10 interi:
[1]:57
[2]:89
[3]:445
[4]:668
[5]:110
[6]:5
[7]:6546
[8]:68
[9]:22
[10]:14
```

2.Dopo di che stampa il vettore "disordinato".

```
Il vettore inserito e':
[1]: 57
[2]: 89
[3]: 445
[4]: 668
[5]: 110
[6]: 5
[7]: 6546
[8]: 68
[9]: 22
[10]: 14
```

3. Successivamente ordina e stampa gli elementi del vettore in modo crescente.

```
Il vettore ordinato e':
[1]:5
[2]:14
[3]:22
[4]:57
[5]:68
[6]:89
[7]:110
[8]:445
[9]:668
[10]:6546
```

4.1Le nostre ipotesi si sono rivelate corrette.

2.Ora dobbiamo andare a modificare il programma per ottenere un errore di segmentazione.

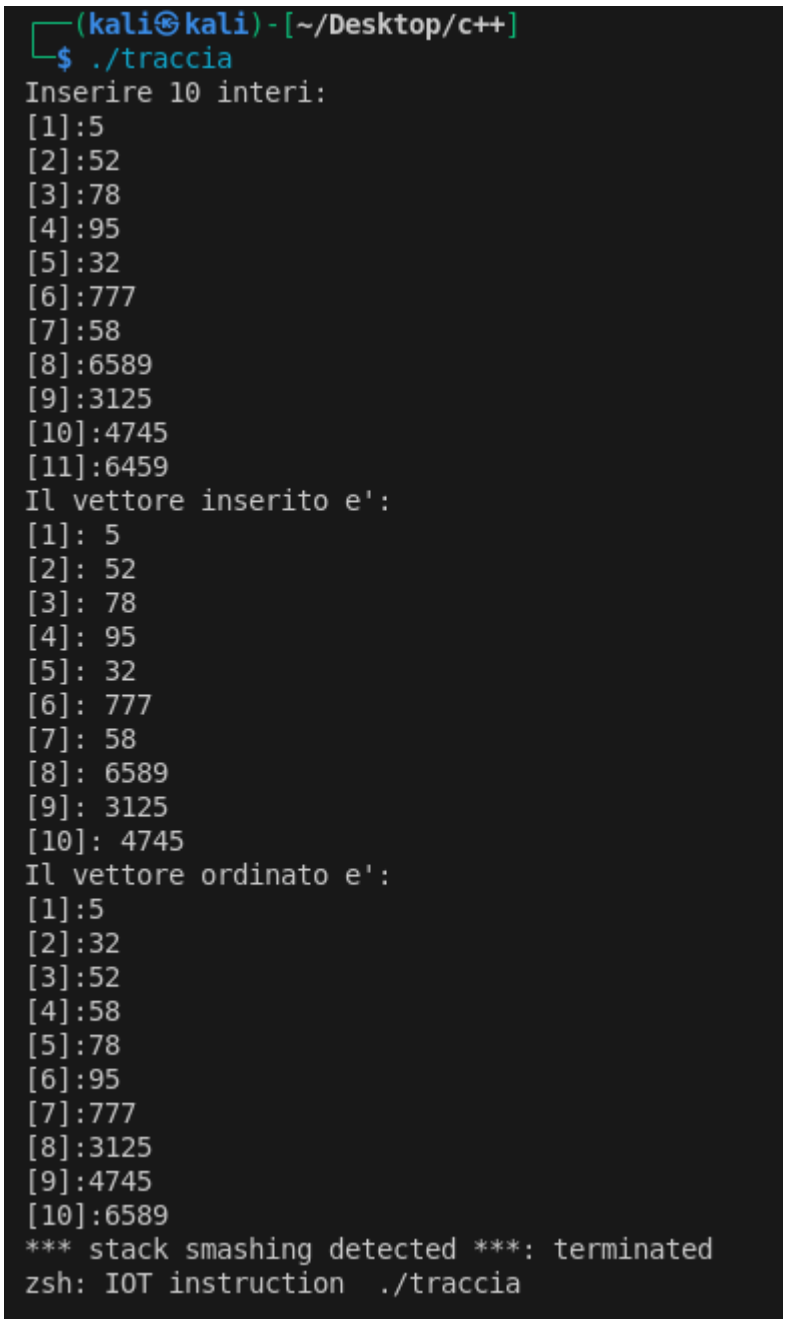
2.1 Per prima cosa andiamo a modificare la condizione da **i < 10 a i <= 10**, questo nel primo ciclo for.

```c
#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]);
    }
```

2.2 Successivamente andiamo a disabilitare le protezioni del compilatore **GCC,** questo perchè contiene misure di sicurezza per evitare il **BOF.**

```
┌──(kali㉿kali)-[~/Desktop/c++]
└─$ gcc -fstack-protector -fstack-protector-all -O0 -o traccia traccia.c
```

Per farlo andiamo ad utilizzare questo comando, dove **“-fstack-protector”** è la protezione dello **stack** contro il **BOF,** e aggiungendoci il comando **-all** la rafforza e la applica a tutte le funzioni. Adesso per disattivare queste protezioni usiamo il comando **-O0**, che disattiva tutte le ottimizzazioni del compilatore.



2.3 Come possiamo notare una volta disattivate le protezioni, viene stampato **l'errore di segmentazione.**

3.BONUS

3.1Nella traccia bonus andremo ad inserire dei **controlli di input** per il nostro codice e **daremo la possibilità all'utente di scegliere quale programma utilizzare**: quello che va in errore oppure quello corretto.

Per il controllo dell'input abbiamo inserito:

1.Un **range** di numeri da poter inserire (da -1000 a 1000)

2.La verica del **tipo** di dato inserito (nel nostro caso un numero intero)

3.Abbiamo inoltre inserito la variabile **check** per confermare che i controlli siano stati effettuati

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

for ( i = 0 ; i < 10 ; i++) {
    int c= i+1;
    do {
        check = 0;
        printf("[%d]:", c);
        if(scanf ("%d", &num) != 1) { //verifica inserimento di un numero
            printf("Errore: Inserire solo numeri interi\n");
            while(getchar() != '\n'); //Pulizia buffer
        } else if (num < -1000 || num > 1000){ //range di numeri da inserire
            printf("Errore: Inserire numeri tra -1000 e 1000 \n" );
        } else {
            check = 1; //conferma di tutti i controlli
        }
    } while (check == 0);
    vector[i] = num;
}
```

Ecco **gli output** del nostro codice

```
┌──(kali㉿kali)-[~/Desktop/c++]
└─$ ./traccia1
Inserire 10 interi:
[1]:a
Errore: Inserire solo numeri interi
```

```
┌──(kali㉿kali)-[~/Desktop/c++]
└─$ ./traccia1
Inserire 10 interi:
[1]:1111
Errore: Inserire numeri tra -1000 e 1000
[1]:
```

3.2 Per la seconda parte del bonus abbiamo inserito la scelta del tipo di codice

```c
do {
    printf("Inserisci quale programma vuoi eseguire:\n1. Programma corretto;\n2. Programma senza controlli con errore. \n");
    scanf("%d", &choice);
} while(choice != 1 && choice != 2);

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

if (choice == 1) {
    for ( i = 0 ; i < 10 ; i++) {
        int c= i+1;
        do {
            check = 0;
            printf("[%d]:", c);
            if(scanf ("%d", &num) != 1) {//verifica inserimento di un numero
                printf("Errore: Inserire solo numeri interi\n");
                while(getchar() != '\n'); //Pulizia buffer
            } else if (num < -1000 || num > 1000){//range di numeri
                printf("Errore: Inserire numeri tra -1000 e 1000");
            } else {
                check = 1;//conferma di tutti i controlli
            }
        } while (check == 0);
        vector[i] = num;
    }
} else {
    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");
    }
```

La scelta avviene tramite un menu in cui l'utente ha due scelte. Scegliendo la prima utilizza il codice corretto, scegliendo la secondo utilizza il codice **difettato**.

```
┌──(kali㉿kali)-[~/Desktop/c++]
└─$ ./traccia1
Inserire 10 interi:
Inserisci quale programma vuoi eseguire:
1. Programma corretto;
2. Programma senza controlli con errore.
```

Avviando il programma l'utente avrà la possibilità di scegliere fin dall'inizio quale programma utilizzare.