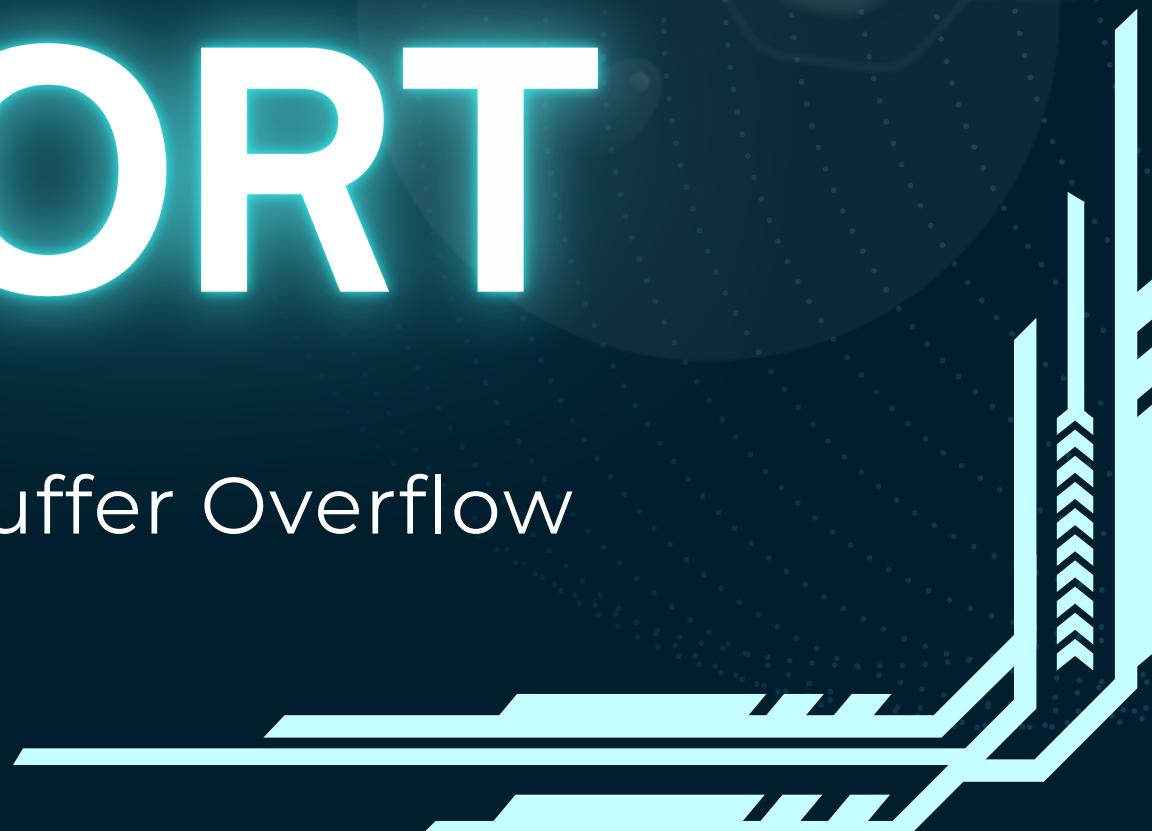




# REPORT

System Exploit Buffer Overflow



## Obiettivo

L'obiettivo di questa esercitazione è quello di comprendere in modo pratico le dinamiche di un attacco Buffer Overflow, simulando un scenario reale di come queste modifiche abbiano causato il superamento dei limiti di un buffer.

## Cos' è il Buffer Overflow

È una vulnerabilità che prende di mira la memoria buffer all'interno delle macchine. Quando il programma cerca di scrivere più dati di quelli che il buffer può contenere, i dati in eccesso possono sovrascrivere la memoria adiacente.

## Cos' è il Buffer

La memoria buffer è una porzione di memoria utilizzata per memorizzare temporaneamente i dati mentre vengono trasferiti da un luogo all'altro.

# Descrizione ed esecuzione del codice in C

```
1 #include <stdio.h>
2
3 int main ()
4 {
5     int vector [10], i, j, k;
6     int swap_var;
7
8     printf ("Inserire 10 interi:\n");
9
10    for ( i = 0 ; i < 10 ; i++)
11    {
12        int c= i+1;
13        printf("[%d]: ", c);
14        scanf ("%d", &vector[i]);
15    }
16
17
18
19    printf ("Il vettore inserito e':\n");
20    for ( i = 0 ; i < 10 ; i++)
21    {
22        int t= i+1;
23        printf("[%d]: %d", t, vector[i]);
24        printf("\n");
25    }
26
27
28    for ( j = 0 ; j < 10 - 1; j++)
29    {
30        for ( k = 0 ; k < 10 - j - 1; k++)
31        {
32            if (vector[k] > vector[k+1])
33            {
34                swap_var=vector[k];
35                vector[k]=vector[k+1];
36                vector[k+1]=swap_var;
37            }
38        }
39    }
40    printf("Il vettore ordinato e':\n");
41    for (j = 0; j < 10; j++)
42    {
43        int g = j+1;
44        printf("[%d]: ", g);
45        printf("%d\n", vector[j]);
46    }
47
48    return 0;
49
50
51 }
```

Da una lettura veloce del codice, si può dedurre, dalle variabili e dai cicli utilizzati, che il programma mette in ordine dieci numeri interi inseriti dall'utente.

```
Il vettore inserito e':
[1]: 10
[2]: 50
[3]: 30
[4]: 70
[5]: 60
[6]: 80
[7]: 20
[8]: 40
[9]: 100
[10]: 90
Il vettore ordinato e':
[1]:10
[2]:20
[3]:30
[4]:40
[5]:50
[6]:60
[7]:70
[8]:80
[9]:90
[10]:100
==== Code Execution Successful ===
```

Una volta eseguito il programma, si ha la conferma delle precedenti deduzioni.

Come rappresentato nell'immagine, si può notare che i vettori inseriti in modo casuale dall'utente vengono riordinati dal valore più basso al valore più alto.

## Modifica del codice

Per indurre un Buffer Overflow, è necessario modificare il programma in modo tale da dare all'utente la possibilità di inserire più vettori di quelli che la memoria può processare. Questo causerà, dunque, un eccesso di dati nella cella stack che porterà ad un malfunzionamento del codice, restituendo l'errore “Segmentation Fault”.

Nello specifico le modifiche da apportare al codice saranno le seguenti:



Riga 5, sarà modificato il valore “10” col numero “5”

```
5 int vector [10], i, j, k
```



Riga 11, sarà modificato il valore “10” col numero “30”

```
11 for ( i = 0 ; i < 10 ; i++)
```

# Risultato

Così facendo il codice restituirà l'errore come nell'immagine seguente:

The screenshot shows a code editor interface with a dark theme. On the left, the code file 'main.c' is displayed with line numbers from 1 to 32. Lines 5 and 11 have magenta boxes around their respective '5' and '30'. The 'Run' button is highlighted in blue. On the right, the 'Output' tab shows the program's execution. It prints 'Il vettore inserito è:' followed by a list of integers from 1 to 10. Then it prints 'Il vettore ordinato è:' followed by the same list. Finally, it reaches line 32, where it prints 'Segmentation fault'.

```
main.c
1 #include <stdio.h>
2
3 int main () {
4
5 int vector[5], i, j, k;
6 int swap_var;
7
8
9 printf ("Inserire 10 interi:\n");
10
11 for ( i = 0 ; i < 30; i++)
12 {
13     int c= i+1;
14     printf("%d:", c);
15     scanf ("%d", &vector[i]);
16 }
17
18
19 printf ("Il vettore inserito è:\n");
20 for ( i = 0 ; i < 10 ; i++)
21 {
22     int t= i+1;
23     printf("%d: %d", t, vector[i]);
24     printf("\n");
25 }
26
27
28 for (j = 0 ; j < 10 ; j++)
29 {
30     for (k = 0 ; k < 10 - j - 1; k++)
31     {
32         if (vector[k] > vector[k+1])
33     }
34 }
```

Output

```
[21]:20
[22]:21
[23]:22
[24]:23
[25]:24
[26]:25
[27]:26
[28]:27
[29]:28
[30]:29
Il vettore inserito è:
[1]: 1
[2]: 2
[3]: 3
[4]: 4
[5]: 5
[6]: 30
[7]: 7
[8]: 8
[9]: 9
[10]: 10
Il vettore ordinato è:
[1]:1
[2]:2
[3]:3
[4]:4
[5]:5
[6]:10
[7]:30
[8]:30
[9]:9
[10]:0
Segmentation fault
```

## Considerazioni finali

IL Buffer Overflow può esser volontario o involontario:

### **Volontario**

In questo caso, l'attaccante potrebbe sfruttare questa tipologia d'attacco per eseguire codice malevolo, per effettuare un'escalation dei privilegi e accesso non autorizzato a dati sensibili.

### **Involontario**

In questo caso non è il risultato di un attacco intenzionale, ma piuttosto di errori di programmazione o cattiva gestione della memoria da parte degli sviluppatori.