[CONSEGNA SETTIMANA 7 LEZIONE 4]

Buffer Overflow

Una delle principali vulnerabilità che possiamo trovare all'interno di un codice (nel nostro caso in linguaggio C) è il ***buffer overflow****, che avviene quando non è stato implementato un controllo nell'input dato dall'utente.*

Nel nostro caso ci ritroveremo davanti una situazione particolare, chiamata ***segmentation fault***, dove il programma tenta di scrivere su una parte di memoria a cui non ha accesso.

```c
#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;
}


```

Possiamo notare che nel codice qui sopra rappresentato, non vi sono controlli per quanto riguarda l'input, nè per **data type**, ne per **grandezza**. La variabile nominata buffer contiene massimo 10 caratteri, cosa succede se ne inseriamo una quantità maggiore?

```
┌──(kali㉿kali)-[~/Desktop]
└─$ cd "/home/kali/Desktop/" && gcc bof.c -o bof && "/home/kali/Desktop/"bof
Si prega di inserire il nome utente: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
Nome utente inserito: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
zsh: segmentation fault  "/home/kali/Desktop/"bof
```

Dopo aver eseguito il codice ho inserito una stringa di testo lunga circa 30 caratteri, nonostante il programma abbia correttamente riportato la stringa inserita, ha anche segnalato un errore, ovvero il *segmentation fault*.

Nel caso avessimo modificato la grandezza del vettore, nel nostro caso a 30, il programma non avrebbe avuto problemi a leggere l'intera stringa e salvarla in memoria. Sarebbe servito un numero ben maggiore per poter riprodurre l'errore di buffer overflow.