CONSEGNA 7 MARZO S7L4

L'obbiettivo è quello di scatenare una situazione di errore particolare chiamata "segmentazione fault", ovvero un errore di memoria che si presenta quando un programma cerca inavvertitamente di scrivere una posizione di memoria dove non gli è permesso scrive. Rientriamo nella classe di vulnerabilità chiamate Buffer Overflow.

Userò un esempio di codice C volutamente vulnerabile (Fig.1).

```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;

}
```

Fig.1

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire il nome utente:oliviero
Nome utente inserito: oliviero

┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire il nome utente:ciaomichiamojohnwayne
Nome utente inserito: ciaomichiamojohnwayne
zsh: segmentation fault  ./BOF

┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire il nome utente:ACOHOGLORIEUF
Nome utente inserito: ACOHOGLORIEUF

┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire il nome utente:GHFJDNGHFJVNDHFGTYRJEHSKCJGNDFHWICNFH
Nome utente inserito: GHFJDNGHFJVNDHFGTYRJEHSKCJGNDFHWICNFH
zsh: segmentation fault  ./BOF
```

Fig.2

Dopo aver scritto il codice ed averlo salvato nel file BOF.c, lanciamo il codice e possiamo vedere che se inserisco più caratteri di quelli accettati dal Buffer, ci ritorna un segmentation fault. Se per esempio inserisco 20 caratteri (ed il buffer accetta 10 caratteri), i caratteri in eccesso sovrascriveranno aree di memoria accessibili. (Fig.2)

Per concludere, il buffer overflow è una vulnerabilità di sicurezza comune che si verifica quando un programma scrive più dati del previsto in un buffer di memoria, sovraffluendo il buffer stesso e potenzialmente corrompendo dati o comportamenti del programma. Nel codice in Fig.1, il buffer è definito come char buffer[10];, il che significa che può contenere fino a 10 caratteri, inclusi il carattere terminatore nullo \0.
Tuttavia, quando utilizzi scanf per leggere una stringa, non viene specificato un limite massimo alla lunghezza della stringa che può essere inserita dall'utente. Questo significa che un utente malintenzionato potrebbe inserire una stringa più lunga di 10 caratteri, causando un overflow del buffer.

Ad esempio, se un utente inserisce una stringa di 15 caratteri, scanf cercherà di memorizzare tutti i 15 caratteri nel buffer di 10 caratteri, sovrascrivendo così la memoria circostante, inclusi altri dati importanti o persino il flusso di esecuzione del programma. Questo può portare a comportamenti imprevisti, crash del programma o, in alcuni casi, essere sfruttato da attaccanti per eseguire codice dannoso o prendere il controllo del programma.

Per mitigare il rischio di buffer overflow, è importante limitare la quantità di dati che scanf può leggere, ad esempio utilizzando %9s anziché %s. Inoltre, è possibile utilizzare funzioni più sicure come fgets che consentono di specificare la dimensione massima del buffer di input, riducendo così il rischio di overflow del buffer.