Floriana Feminò

08/12/2022

BUFFER OVERFLOW

Il Buffer Overflow è una vulnerabilità causata da una mancanza di controllo dei limiti dei buffer che accettano un input dall'utente: un input maggiore della dimensione del buffer finisce per sovrascrivere il contenuto degli indirizzi di memoria adiacenti, modificando il contenuto delle variabili di un programma. Se il contenuto viene modificato con un indirizzo che punta ad una porzione di memoria contenente codice inserito dall'attaccante, questi potrà far eseguire al programma codice arbitrario.

Per testare la vulnerabilità in oggetto, prenderemo in esame un semplice codice in C che prende in input un buffer di massimo 10 caratteri:

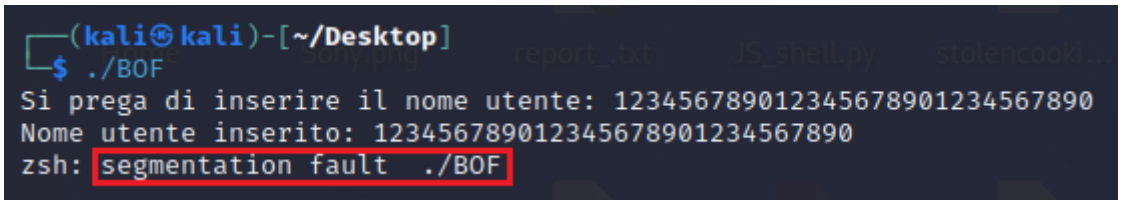
```
 GNU nano 6.4                                                    BOF.c
#include <stdio.h>

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;

}
```

Dopo averlo compilato, verifichiamo in primis il comportamento del programma dando in input un numero di caratteri nel range accettato:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ gcc BOF.c -o BOF
```

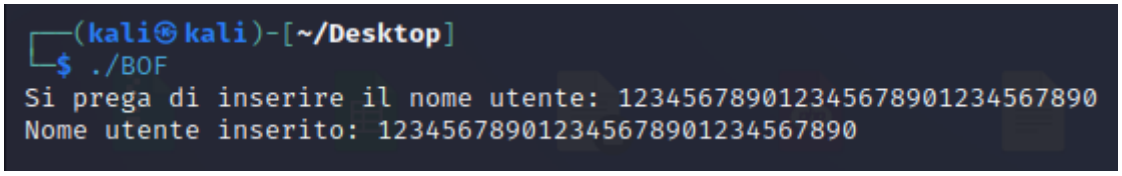
```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire il nome utente: 1234567890
Nome utente inserito: 1234567890
```

Come si vede, inserendo 10 caratteri il programma viene eseguito normalmente e senza errori. provando a immettere un input di caratteri molto maggiore al limite impostato dal programma. Proviamo con 30:

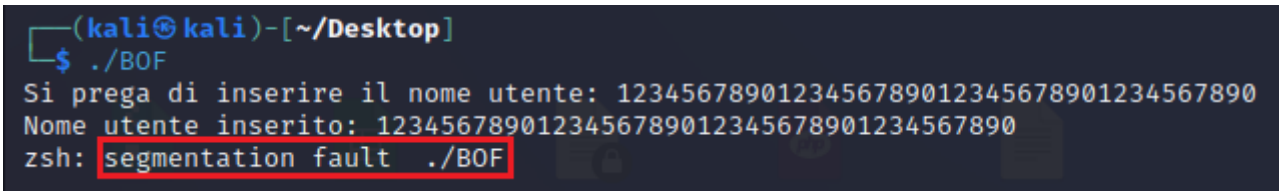


Otteniamo un errore di segmentazione (*segmentation fault*): questo tipo di circostanza si verifica quando un programma tenta di scrivere contenuti su una porzione di memoria al quale non ha accesso: abbiamo inserito 30 caratteri in un buffer che può contenerne 10 e di conseguenza alcuni caratteri stanno sovrascrivendo aree di memoria inaccessibili.

Ovviamente tutto cambia modificando il valore del buffer: scegliamo un range massimo di 30 caratteri, ricompiliamo il programma e facciamo alcuni test. Inseriamo 30 caratteri in input:



Come da aspettative, il programma viene eseguito normalmente. Adesso proviamo ad inserire invece 40 caratteri:



In questo caso, abbiamo ottenuto un errore di segmentazione.