Buffer Overflow

Introduzione

L'obiettivo dell'esercizio è comprendere il funzionamento di una vulnerabilità di tipo **Buffer Overflow (BOF)**, causata dalla mancata validazione dell'input utente. Attraverso un semplice programma scritto in linguaggio C, viene mostrato come un input più lungo del buffer allocato possa generare un errore di memoria chiamato **segmentation fault** .

Creazione e compilazione del programma

È stato creato un file BOF.c contenente un programma volutamente vulnerabile, nel quale viene dichiarato un buffer di dimensione fissa e viene utilizzata la funzione scanf("%s", buffer) senza alcun controllo sulla lunghezza dell'input.

Il programma è stato successivamente compilato tramite il compilatore **gcc** senza riscontrare errori.

```
 GNU nano 8.6                    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;

}
```

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

┌──(pireddone㉿kali)-[~/Desktop]
└─$
```

Esecuzione e generazione del Buffer Overflow

Durante l'esecuzione del programma, inserendo un input di lunghezza ridotta, il programma funziona correttamente e stampa il valore inserito.

Inserendo invece una stringa molto più lunga rispetto alla dimensione del buffer, il programma genera un errore di **segmentation fault**, dimostrando la presenza di una vulnerabilità di tipo Buffer Overflow, causata dalla sovrascrittura di aree di memoria non consentite.

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

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

Modifica del buffer e considerazioni

Successivamente, il programma è stato modificato aumentando la dimensione del buffer da 10 a 30 caratteri.

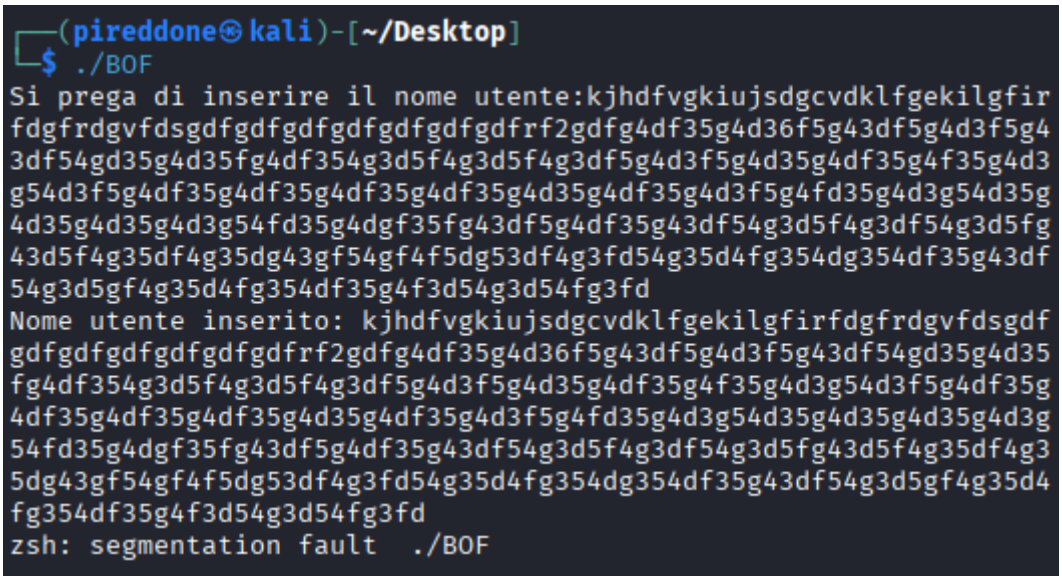
In questo caso, inserendo una stringa di circa 30 caratteri il programma non va in errore, mentre fornendo un input molto più lungo viene nuovamente generato un **segmentation fault**.

Questo dimostra che **aumentare la dimensione del buffer non elimina la vulnerabilità**, ma ne sposta solamente il limite. L'assenza di controlli sull'input rende il programma comunque vulnerabile a Buffer Overflow.

```
  GNU nano 8.6                    BOF.c *
#include <stdio.h>

int main() {
        char buffer[30];

        printf("Si prega di inserire il nome utente:");
        scanf("%s", buffer);

        printf("Nome utente inserito: %s\n", buffer);

        return 0;

}
```

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

┌──(pireddone㉿kali)-[~/Desktop]
└─$
```

```
┌──(pireddone㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire il nome utente:123456789098765432123456789098
Nome utente inserito: 12345678909876543212345678909
```



Conclusione

L'esercizio evidenzia come una gestione non sicura dell'input possa portare a gravi problemi di sicurezza.

La prevenzione dei Buffer Overflow richiede l'adozione di controlli sulla lunghezza dei dati inseriti e l'utilizzo di funzioni più sicure per la gestione degli input utente.