CONSEGNA W17D4

Traccia: Il buffer overflow è una vulnerabilità conseguenza di una mancanza di controllo dei limiti dei buffer che accettano input utente. Vedremo un esempio di codice in C volutamente vulnerabile ai BOF, e come scatenare una situazione di errore particolare chiamata «segmentation fault», ovvero un errore di memoria che si presenta quando un programma cerca inavvertitamente di scrivere su una posizione di memoria dove non gli è permesso scrivere (come può essere ad esempio una posizione di memoria dedicata a funzioni del sistema operativo).

Facoltativo: Aggiungete nel programma precedente dei controlli di sicurezza per prevenire il BOF e sanitizzare l'input.

Come richiesto dalla traccia andiamo sul terminale entriamo sul terminale ed andiamo, tramite il comando cd, sul Desktop. Dopo creiamo un file inserendo sul terminale il comando "nano BOF.c" dove inseriamo un piccolo codice come nella figura seguente

```
  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;
}
```

Dopo aver avviato il programma lo testiamo vedendo che, come da codice, il limite di caratteri è di 10, infatti se inseriamo 30 caratteri il programma ci ritorna un errore, «segmentation fault», ovvero errore di segmentazione. L'errore di segmentazione avviene quando un programma, come abbiamo detto in precedenza, tenta di scrivere contenuti su una porzione di memoria alla quale non ha accesso. Questo è un chiaro esempio di BOF, abbiamo inserito più caratteri in un buffer che ne può contenere solamente 10 e di conseguenza alcuni caratteri stanno sovrascrivendo aree di memorie inaccessibili.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire il nome utente:test1
Nome utente inserito: test1

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

Andiamo dunque a modificare il codice scritto in precedenza modificando il numero tra le parentesi quadre aumentando il limite a 30 caratteri e testiamo le modifiche dopo averle salvate e rese attive. Vediamo i risultati di tali modifiche nella figura sottostante

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

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

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

PARTE FACOLTATIVA

Per effettuare la parte facoltativa dell'esercizio vediamo che dobbiamo modificare di nuovo il codice ma questa volta modificando la parte relativa allo scanf questo permetterà al programma di leggere e usare solo il numero di lettere che inseriamo tra le virgolette come nella figura seguente.

```
  GNU nano 8.6
#include <stdio.h>

int main () {

char buffer [30];

printf ("Si prega di inserire il nome utente:");
scanf ("%10s", buffer);

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

return 0;
}
```

Vediamo ora come risponde alle modifiche il "programma" creato

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

Con questa esercitazione mostriamo una vulnerabilità critica nota nei programmi scritti in C, questa vulnerabilità prende il nome di "buffer overflow". Questa vulnerabilità è un errore che si verifica quando un programma scrive dati oltre i limiti di un buffer sovrascrivendo in questo modo aree di memoria non destinate a quell'uso. Questa vulnerabilità può essere sfruttata da un hacker per eseguire codice al fine di ottenere privilegi o compromettere il sistema. La parte facoltativa ci mostra come una piccola modifica nella gestione dell'input possa evitare la compromissione della sicurezza di un programma rendendo in questo modo il codice più robusto riducendo la superficie d'attacco.