Giorno 4: BOF EPICODE

Giovanni Pisapia

CREAZIONE CODICE C

- Apriamo il terminale con il comando sudo nano BOF.c creiamo un nuovo file
- Scriviamo il codice come da traccia
- Salviamo il file con Ctrl + x.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo nano BOF.c
[sudo] password for kali:
```

```c
#include <stuidio.h>


int main() {

char buffer[10];

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

printf("nome utente inseirit: %s\n", buffer);


return 0;
}
```

PROVA BOF

- Una volta creato il file lo compiliamo con il comando gcc -g BOF.c -o BOF
- Una volta compilato il file lo avviamo con ./BOF
- Proviamo ad inserire un nome utente più lungo di 10 caratteri è ci viene mostrato l'errore di segmention fault ,che ci avvisa che stiamo provando ad accedere a una parte della memoria non valida o non allocata .
- Per avere prova visiva che il buffer over flow si avvenuto creiamo un nuovo codice inserendo una nuova variabile
- Come possiamo notare la variabile buffer verrà sovrascritta dalla variabile buffer 2, nel momento del print dell' input utente

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

┌──(kali㉿kali)-[~/Desktop]
└─$ BOF
BOF: command not found

┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire un nome utente:GIOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
nome utente inseirit: GIOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOOO
zsh: segmentation fault  ./BOF
```

```c
#include <stdio.h>


int main() {

char buffer[10];
char buffer2[10];


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

printf("Si prega di inserire il cognome:");
scanf("%s",buffer2);



printf("nome inseirito: %s\n", buffer);
printf("cognome inseirito: %s\n", buffer2);


return 0;
}
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ./BOF
Si prega di inserire il nome :ciaooooooooooooooo
Si prega di inserire il cognome:ciaooooooosdas
nome inseirito: osdas
cognome inseirito: ciaooooooosdas
zsh: segmentation fault  ./BOF
```

MITIGATION BFO

Per mitigare questa vulnerabilità creiamo un nuovo codice :

- Inseriamo due costanti per specificare la dimensione massima dei buffer utilizzati per salvare nome e cognome.
- Leggiamo l'input dell'utente con fgets
- Controlliamo la lunghezza con la funzione strlen
- E infine se l'utente inserisce più caratteri di quelli consenti il programma si chiude in automatico.

```
 GNU nano 7.2
#include <stdio.h>
#include <string.h>

#define DIMENSIONE_BUFFER_1 20
#define DIMENSIONE_BUFFER_2 20

int main() {
    char buffer[DIMENSIONE_BUFFER_1];
    char secondBuffer[DIMENSIONE_BUFFER_2];

    printf("Si prega di inserire il tuo nome: ");
    fgets(buffer, sizeof(buffer), stdin);
    buffer[strcspn(buffer, "\n")] = '\0';  // Rimuove il newline finale

    if (strlen(buffer) >= DIMENSIONE_BUFFER_1 - 1) {
        printf("Hai inserito troppi caratteri nel nome. Il programma termina.\n");
        return 0;  // Termina il programma
    }

    printf("Si prega di inserire il tuo cognome: ");
    fgets(secondBuffer, sizeof(secondBuffer), stdin);
    secondBuffer[strcspn(secondBuffer, "\n")] = '\0';  // Rimuove il newline finale

    if (strlen(secondBuffer) >= DIMENSIONE_BUFFER_2 - 1) {
        printf("Hai inserito troppi caratteri nel cognome. Il programma termina.\n");
        return 0;  // Termina il programma
    }

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

    return 0;
}
```

```
Si prega di inserire il tuo nome: giovanni
Si prega di inserire il tuo cognome: ciaooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
Hai inserito troppi caratteri nel cognome. Il programma termina.
```